QmeQ 1.0: An open-source Python package for calculations of transport through quantum dot devices

Kiršanskas, Gediminas; Pedersen, Jonas Nyvold; Karlström, Olov; Leijnse, Martin Christian; Wacker, Andreas

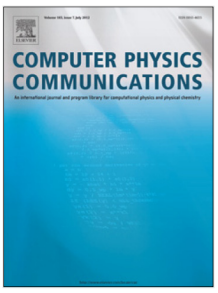
Published in:
Computer Physics Communications

Link to article, DOI:
10.1016/j.cpc.2017.07.024

Publication date:
2017

Document Version
Peer reviewed version

Accepted Manuscript



`QmeQ` 1.0: An open-source `Python` package for calculations of transport through quantum dot devices

Gediminas Kiršanskas, Jonas Nyvold Pedersen, Olov Karlström, Martin Leijnse, Andreas Wacker

PII: S0010-4655(17)30251-5
DOI: http://dx.doi.org/10.1016/j.cpc.2017.07.024
Reference: COMPHY 6291

To appear in: *Computer Physics Communications*

Received date : 14 March 2017
Revised date : 15 June 2017
Accepted date : 28 July 2017

Please cite this article as: G. Kiršanskas, J.N. Pedersen, O. Karlström, M. Leijnse, A. Wacker, `QmeQ` 1.0: An open-source `Python` package for calculations of transport through quantum dot devices, *Computer Physics Communications* (2017), http://dx.doi.org/10.1016/j.cpc.2017.07.024

QmeQ 1.0: An open-source Python package for calculations of transport through quantum dot devices

Gediminas Kiršanskas[a], Jonas Nyvold Pedersen[c], Olov Karlström[a], Martin Leijnse[b], Andreas Wacker[a]

[a]*Mathematical Physics and NanoLund, Lund University, Box 118, 22100 Lund, Sweden*
[b]*Solid State Physics and NanoLund, Lund University, Box 118, 221 00 Lund, Sweden*
[c]*Department of Micro- and Nanotechnology, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark*

Abstract

QmeQ is an open-source Python package for numerical modeling of transport through quantum dot devices with strong electron-electron interactions using various approximate master equation approaches. The package provides a framework for calculating stationary particle or energy currents driven by differences in chemical potentials or temperatures between the leads which are tunnel coupled to the quantum dots. The electronic structures of the quantum dots are described by their single-particle states and the Coulomb matrix elements between the states. When transport is treated perturbatively to lowest order in the tunneling couplings, the possible approaches are *Pauli* (classical), *first-order Redfield*, and *first-order von Neumann* master equations, and a particular form of the *Lindblad* equation. When all processes involving two-particle excitations in the leads are of interest, the *second-order von Neumann* approach can be applied. All these approaches are implemented in QmeQ. We here give an overview of the basic structure of the package, give examples of transport calculations, and outline the range of applicability of the different approximate approaches.

Keywords: Open quantum systems, Quantum dots, Anderson-type model, Coulomb blockade, Python

Program summary

Program Title: QmeQ
Program Files doi: http://dx.doi.org/10.17632/8687mrhgg9.1
Licensing provisions: BSD 2-Clause
Programming language: Python
External libraries: NumPy, SciPy, Cython
Nature of problem: Calculation of stationary state currents through quantum dots tunnel coupled to leads.
Solution method: Exact diagonalisation of the quantum dot Hamiltonian for a given set of single particle states and Coulomb matrix elements. Numerical solution of the stationary-state master equation for a given approximate approach.
Restrictions: Depending on the approximate approach the temperature needs to be sufficiently large compared to the coupling strength for the approach to be valid.

1. Introduction

Quantum dot devices are usually made of a nanostructure or a molecule coupled to leads [1, 2, 3]. A vast amount of experiments have been performed using quantum dots, where with bias spectroscopy [4] or thermoelectric measurements [5], it is possible to obtain the energy level structure of the quantum dot, which is important for understanding and predicting the device behavior. Such a device is also a useful tool for studying the fundamental physics of open-quantum systems in non-equilibrium, where the leads play the role of an environment [6]. Modeling transport through such nanoscale systems is a complicated task, especially when strong Coulomb interaction is present in the quantum dots, as it can lead to phenomena like Coulomb blockade [7, 8], Kondo effect [9, 10, 11], or Pauli blockade [12], just to mention a few. Quantum transport can be calculated using different theoretical methods like scattering-

states numerical-renormalization group [13], non-equilibrium Green's functions [14, 15], and master equation based approaches [6, 16, 17, 18, 19, 20, 21, 22].

We consider here approximate master equation approaches, which were also used to interpret transport experimental data in the regime of weak coupling to the environment [23, 24, 25]. In particular, we address the so-called *Pauli* master equation [6, 26, 27], the *first-order Redfield* approach [6, 28, 29], the *first/second-order von Neumann* approaches [19, 30], and a particular form of the *Lindblad* equation [31, 32], and apply these methods for tunneling models, where the quantum dots can have arbitrary Coulomb interactions [33]. Even though there is a lot of literature on such methods, there is no publicly-available, well-documented package, which is easy to apply for quantum dot model systems. One reason is that depending on parameter regime these methods can fail and a good knowledge of the method is required to know whether to trust the result or not. For example, these methods can violate positivity [6] of the reduced density matrix and lead to large currents flowing against the bias [34]. Nevertheless, we think it is important to have a package where a user can duplicate existing calculations, check the applicability of different methods, or simply discover new kind of physics using different approximate master equations.

In this paper, we describe an open-source package `QmeQ` designed for simulating stationary-state transport through quantum dots (available at `http://github.com/gedaskir/qmeq`). The name `QmeQ` deciphers as ***Q**uantum **m**aster **e**quation* for ***Q**uantum* dot transport calculations. It is written in the programming languages `Python` [35] and `Cython` [36, 37]. In recent years, a large number of `Python`-based scientific computation packages have appeared [38] because of the development of `NumPy` [39] and `SciPy` [40]. These have modules for linear algebra, optimisation, and special functions, which also `QmeQ` relies on. Thus it is convenient to use the same open-source platform based on `Python`, which provides an easy integration between different kind of packages. Additionally, `Python` is a high-level interpreted language, which allows development of an easy-to-read code and easy usage. For some parts of the code we use `Cython` which lets us to compile bottleneck parts of the package to achieve better performance.[1] Furthermore, in our examples for data visualization we use the `matplotlib` package [41], the tutorials are provided using `Jupyter` notebooks [42], and detailed documentation of the code is generated using `Sphinx` [43].

There are already a variety of well-developed open-source `Python` packages dedicated to model quantum mechanical phenomena and transport. One such package is `ASE` (Atomic Simulation Environment), which performs atomistic simulations with the possibility to make transport calculations in molecular junctions using density-functional theory (DFT) and non-equilibrium Green's functions (NEGF) [44]. For numerical calculations on tight-binding models with a focus on quantum transport the `Kwant` package can be used [45]. The simulation of the dynamics of open quantum systems using master equations in Lindblad, Floquet–Markov, or Bloch–Redfield form can be performed using the `QuTiP` (Quantum Toolbox in Python) framework [46, 47]. In contrast to `QuTiP`, we focus on a particular model for the quantum dot device, where Coulomb interaction between particles can be dominating, and allow for a direct comparison between different approximate quantum master equations, including the *second-order von Neumann* approach. For an overview of other transport packages including commercial ones also see Section 4 of Ref. [45]. Lastly, the cotunneling calculations using a T-matrix based approach [27] can be performed using the `humo` package.[2]

The paper is organized as follows. In Section 2 we introduce the model Hamiltonian for the quantum dot device and in Section 3 we describe how such a Hamiltonian is set up in `QmeQ`. In Sections 4, 5, and 6 we give examples of transport calculation through various quantum dot devices, using different approaches (examples scripts are available at `http://github.com/gedaskir/qmeq-examples`). The derivation of the approximate master equations used in `QmeQ` is relegated to Appendices. Throughout the paper our units are such that $\hbar = 1$, $k_B = 1$, $|e| = 1$.

2. The model

The system under consideration consists of a number of quantum dots attached to a number of metallic leads. Such a quantum dot device can be modelled by the following Hamiltonian:

$$H = H_{\text{leads}} + H_{\text{tunneling}} + H_{\text{dot}}, \tag{1}$$

[1] In short, `Cython` converts `Python`-like code with static type definitions into `C` code.

[2] Available at `http://github.com/georglind/humo`. This package is currently rather undocumented but the used methods are well described in Ref. [48]. The calculations can be performed on models of the type described by Eq. (1).

where H_{leads} describes electrons in the leads, H_{dot} describes electrons in the dot, and $H_{\text{tunneling}}$ corresponds to tunneling between the leads and the dot. The leads are described as non-interacting electrons

$$H_{\text{leads}} = \sum_{\alpha k} \varepsilon_{\alpha k} c^\dagger_{\alpha k} c_{\alpha k}, \tag{2}$$

where $c^\dagger_{\alpha k}$ creates an electron in the lead channel α with k representing a continuous quantum number, which refers to a continuum energy. This means that k-sums can be performed by introducing the density of states $\nu(E)$ as $\sum_k f_k \to \int \mathrm{d}E \nu(E) f(E)$. The lead channel α can encompass, for example, an actual source/drain lead label, spin of an electron, etc., depending on actual physical setup under consideration.

For the quantum dot the following rather general many-body Hamiltonian is used

$$H_{\text{dot}} = H_{\text{single}} + H_{\text{Coulomb}}, \tag{3}$$

$$H_{\text{single}} = \sum_i \varepsilon_i d^\dagger_i d_i + \sum_{i \neq j} \Omega_{ij} d^\dagger_i d_j, \tag{4}$$

$$H_{\text{Coulomb}} = \sum_{mnkl} U_{mnkl} d^\dagger_m d^\dagger_n d_k d_l, \quad \text{with } m < n, \tag{5}$$

where $d^\dagger_i$ creates an electron in single-particle orbital i, ε_i is the energy of that orbital, and Ω_{ij} gives the hybridization between single-particle orbitals. Here also the Coulomb interaction U_{mnkl} can be present. We note that the Coulomb interaction takes the form of Eq. (5) (without a factor of $\frac{1}{2}$) for state labels $m < n$. Lastly, the tunneling Hamiltonian is

$$H_{\text{tunneling}} = \sum_{\alpha k i} t_{\alpha k,i} d^\dagger_i c_{\alpha k} + \text{H.c.}, \tag{6}$$

where H.c. denotes Hermitian conjugate of the first term and $t_{\alpha k,i}$ is the tunneling amplitude between the leads and the dot. An important energy scale in the calculations is the tunneling rate defined as

$$\Gamma_{\alpha k,i}(E) = 2\pi \sum_k |t_{\alpha k,i}|^2 \delta(E - \varepsilon_{\alpha k}). \tag{7}$$

In the case of single spinful orbital with on-site interaction U Hamiltonian Eq. (1) is referred to as Anderson-type model [33, 49, 50].

2.1. Approximations

For the calculations in `QmeQ` we make the following important assumption for the model Eq. (1):

1. The leads are thermalized according to a Fermi-Dirac occupation function $f_\alpha(E) = [e^{(E-\mu_\alpha)/T_\alpha} + 1]^{-1}$ with respective temperatures T_α and chemical potentials μ_α.

Additionally, we use the so-called wide-band limit for the leads:

2. The leads have constant density of states $\nu(E) \approx \nu(E_F) = \nu_F$, where the subscript F corresponds to the Fermi level. Then the k-sums are performed as $\sum_k \to \nu_F \int_{-D}^{+D} \mathrm{d}E$, where D denotes bandwidth of the leads. Also the tunneling amplitudes are energy (or k) independent, i.e., $t_{\alpha k,i} \approx t_{\alpha i}$. In this case the tunneling rates become $\Gamma_{\alpha i} = 2\pi \nu_F |t_{\alpha i}|^2$.

Approximation **2.** is made for convenience to simplify the integrals and not to clutter the `QmeQ` code with the specification of energy-dependent functions. The wide-band limit can be easily relaxed. On the other hand, this is often a good approximation for metallic leads.

2.2. Many-body eigenstates and the reduced density matrix

In QmeQ the quantum dot Hamiltonian (3) is constructed in a Fock basis and diagonalized exactly to obtain the many-body eigenstates $|a\rangle$,

$$H_{\text{dot}} = \sum_a E_a |a\rangle\langle a|. \tag{8}$$

We note that a fermionic many-body Hamiltonian is constructed efficiently using Lin tables (for more details see Refs. [48, 51]). The tunneling Hamiltonian is expressed in this many-body eigenbasis as

$$H_{\text{tunneling}} = \sum_{ab,\alpha k} T_{ba,\alpha} |b\rangle\langle a| c_{\alpha k} + \text{H.c.}, \quad T_{ba,\alpha} = \sum_i t_{\alpha i} \langle b| d_i^\dagger |a\rangle, \tag{9}$$

where $T_{ba,\alpha}$ represents many-particle tunneling amplitudes. Here we used the *letter convention*: if more than one state enters an equation, then the position of the letter in the alphabet follows the particle number (for example $N_b = N_a + 1$, $N_c = N_a + 2$, $N_{a'} = N_a$). In such a way the sum $\sum_{bc}$ restricts to those combinations, where $N_c = N_b + 1$. The approximate quantum master equations are constructed in the many-body eigenbasis of the quantum dot and are discussed in Appendices.

The central quantity in the approximate master equations is the reduced density matrix, which is defined as:

$$\Phi^{[0]}_{bb'} = \sum_g \langle bg|\rho|b'g\rangle, \tag{10}$$

with ρ being the full density matrix operator of the system. Here g represents the many-body eigenstates of the lead Hamiltonian H_{leads} (2) with arbitrary occupations of the lead states $k\alpha$. Then a basis of the combined lead and dot system is given by the product states $|bg\rangle = |b\rangle \otimes |g\rangle$.

2.3. Calculation of currents

In QmeQ the current emanating from a particular lead channel α is calculated for a stationary state. The particle current leaving the lead channel α is defined as

$$I_\alpha = -\frac{\partial}{\partial t}\langle N_\alpha\rangle = -i\langle[H, N_\alpha]\rangle, \quad \text{with} \quad N_\alpha = \sum_k c^\dagger_{\alpha k} c_{\alpha k}, \tag{11}$$

and the energy current as

$$\dot{E}_\alpha = -\frac{\partial}{\partial t}\langle H_\alpha\rangle = -i\langle[H, H_\alpha]\rangle, \quad \text{with} \quad H_\alpha = \sum_k \varepsilon_{\alpha k} c^\dagger_{\alpha k} c_{\alpha k}. \tag{12}$$

We introduce the following energy-resolved particle-current amplitudes:

$$\Phi^{[1]}_{cb,\alpha k} = \sum_g \langle cg - \alpha k|\rho|bg\rangle(-1)^{N_b}, \quad \text{with} \quad |bg - \alpha k\rangle = |b\rangle \otimes c_{\alpha k}|g\rangle, \tag{13}$$

where N_b is the number of particles in the many-body state $|b\rangle$. We can express the particle and energy currents as

$$I_\alpha = -2\sum_{k,cb} \text{Im}[T_{bc,\alpha}\Phi^{[1]}_{cb,\alpha k}], \tag{14}$$

$$\dot{E}_\alpha = -2\sum_{k,cb} \varepsilon_{\alpha k}\,\text{Im}[T_{bc,\alpha}\Phi^{[1]}_{cb,\alpha k}]. \tag{15}$$

Lastly, the heat current emanating from the lead channel α can be defined as

$$\dot{Q}_\alpha = \dot{E}_\alpha - \mu_\alpha I_\alpha. \tag{16}$$

The above definition of the heat current corresponds to the rate of change of the entropy $\dot{S}_\alpha = \frac{1}{T_\alpha}(\dot{E}_\alpha - \mu_\alpha \dot{N}_\alpha)$.

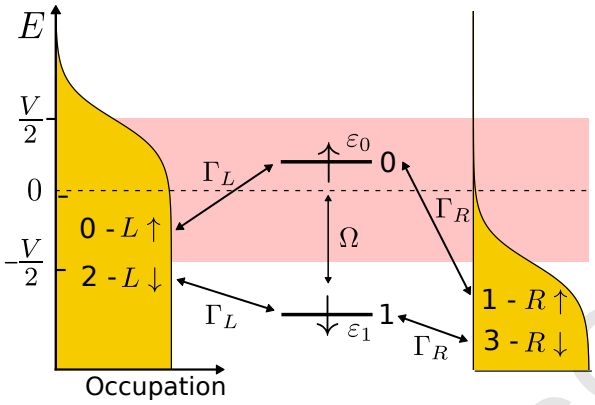


Figure 1: Spinful single orbital coupled to spinful metallic leads, where Γ_L and Γ_R are the tunneling rates to the source (L) and the drain (R) leads, respectively. Numbers represent the labeling of states in the example. Energies of the spin-up and spin-down levels are given by ε_0 and ε_1, respectively. Ω is the hybridisation between the states, which can appear due to presence of spin-orbit coupling (sets a concrete spin quantization axis) and a magnetic field perpendicular to the spin orientation.

3. The `QmeQ` package

The `QmeQ` package diagonalises the quantum dot Hamiltonian (3) exactly and performs various approximate master equation calculations on the system Hamiltonian (1), where tunneling (6) is treated as a perturbation. Installation instructions for `QmeQ` are provided in the INSTALL.md file coming with the source code of the package. We start with a short script giving a minimal example of a quantum dot containing one spinful orbital and on-site charging energy U coupled to source (L) and drain (R) leads as shown in Figure 1:

$$H_{\text{one}} = \sum_{\substack{k;\ell=L,R\\ \sigma=\uparrow,\downarrow}} \varepsilon_{\ell k} c^\dagger_{\ell\sigma k} c_{\ell\sigma k} + \sum_{\ell\sigma k}\left(t_\ell d^\dagger_\sigma c_{\ell\sigma k} + \text{H.c.}\right) + \sum_\sigma \varepsilon_\sigma d^\dagger_\sigma d_\sigma + (\Omega d^\dagger_\uparrow d_\downarrow + \text{H.c.}) + U d^\dagger_\uparrow d^\dagger_\downarrow d_\downarrow d_\uparrow. \tag{17}$$

The script calculating the particle current and the energy current in a single point in the parameter space reads

```
# Prerequisites
import qmeq
from numpy import sqrt, pi, power

# Parameters used (numbers are in arbitrary energy_scale, e.g., meV)
e0, e1, omega, U = 0.0, 0.0, 0.0, 20.0
# dband is bandwidth D of the leads
tempL, tempR, muL, muR, dband= 1.0, 1.0 , 0.2, -0.2, 60.0
gammaL, gammaR = 0.5 , 0.7
tL, tR = sqrt(gammaL/(2*pi)), sqrt(gammaR/(2*pi))

## Hamiltonian, which requires definition

# nsingle is the number of single-particle states
# Here level 0 is spin up; level 1 is spin down
nsingle = 2
hsingle = {(0,0): e0, (1,1): e1, (0,1): omega}
# (0,1,1,0) represents an operator d0^{+}d1^{+}d1^{-}d0^{-}
coulomb = {(0,1,1,0): U}

## Lead and tunneling properties, which requires definition
```

```
# nleads is the number of lead channels
nleads = 4 # for two leads L/R with two spins up/down
# Here   L-up      R-up      L-down    R-down
mulst = {0: muL,   1: muR,   2: muL,   3: muR}
tlst =  {0: tempL, 1: tempR, 2: tempL, 3: tempR}

# The coupling matrix has indices(lead-spin, level)
tleads = {(0,0): tL, (1,0): tR, (2,1): tL, (3,1): tR}

## Construction of the transport system and
## calculation of currents into the system from all 4 channels

# Choice of approximate approach
# For kerntype='Redfield', '1vN', 'Lindblad', or 'Pauli'
system = qmeq.Builder(nsingle, hsingle, coulomb,
                      nleads, tleads, mulst, tlst, dband,
                      kerntype='Pauli')
system.solve()
print('Output is currents in four lead channels')
print('[L-up R-up L-down R-down]\n')
print('Pauli, particle current (in units of energy_scale/hbar):')
print(system.current)
print('Pauli, energy current (in units of energy_scale^2/hbar):')
print(system.energy_current)

# For '2vN' approach
# Here we use 2^12 energy grid points for the leads
kpnt = power(2,12)
system2vN = qmeq.Builder(nsingle, hsingle, coulomb,
                         nleads, tleads, mulst, tlst, dband,
                         kerntype='2vN', kpnt=kpnt)
# 7 iterations is good for temp approx gamma
system2vN.solve(niter=7)
print('\n2vN, particle current (in units of energy_scale/hbar):')
print(system2vN.current)
print('2vN, energy current (in units of energy_scale^2/hbar):')
print(system2vN.energy_current)
```

Here we have calculated the particle current and the energy current from the four lead channels $L\uparrow, R\uparrow, L\downarrow, R\downarrow$ into the dot using *Pauli* master equation and *second-order von Neumann* approach (*2vN*) and we get the following output:

```
Output is currents in four lead channels
[L-up R-up L-down R-down]

Pauli, particle current (in units of energy_scale/hbar):
[ 0.01948779 -0.01948779 0.01948779 -0.01948779]
Pauli, energy current (in units of energy_scale^2/hbar):
[ 1.59585694e-09 -1.59585694e-09 1.59585694e-09 -1.59585694e-09]

2vN, particle current (in units of energy_scale/hbar):
[ 0.01431084 -0.01431084 0.01431084 -0.01431084]
2vN, energy current (in units of energy_scale^2/hbar):
[ 0.00617130 -0.00617129 0.00617130 -0.00617129]
```

We elaborate more on this single spinful orbital example in Section 4. Next we discuss what different parts of the script mean.

From the above example we see that the system is defined using the `Builder` class. For `Builder` to construct a system object, we need to specify at least the following variables:

```
system = qmeq.Builder(nsingle, # Number of single-particle states
                      hsingle, # Single-particle Hamiltonian
                      coulomb, # Coulomb matrix elements
                      nleads,  # Number of lead channels
                      tleads,  # Single-particle tunneling amplitudes
                      mulst,   # Chemical potentials of leads
                      tlst,    # Temperatures of leads
                      dband)   # Bandwidth of leads
```

Additionally, there is a variety of optional arguments, which can be viewed by typing `help(qmeq.Builder)` in the `Python` interpreter. These optional arguments can be set by specifying `Builder(...,` *argument*=*value*`)`.[3]

The variables `nsingle`, `hsingle`, and `coulomb` describe the quantum dot Eq. (3). `nsingle` is the number of single-particle states. Single-particle states are labeled by integers from 0 to `nsingle`−1, and the single-particle Hamiltonian (4) `hsingle` is specified using the `Python` dictionary:

```
hsingle = {(0,0): ε_0, (0,1): Ω_01, ... , (0,nsingle-1): Ω_0,nsingle-1,
           ...
           (i,i): ε_i, (i,i+1): Ω_12, ... , (i,nsingle-1): Ω_i,nsingle-1,
           ...
           (nsingle-1,nsingle-1): Ω_nsingle-1,nsingle-1}
```

In a dictionary it is enough to specify one element like $\Omega_{ij}d_i^\dagger d_j$, because the element $\Omega_{ji}d_j^\dagger d_i$ is determined by complex conjugation $\Omega_{ji} = \Omega_{ij}^*$ and is included automatically in order to get a Hermitian quantum dot Hamiltonian. If an element like `(j,i)` is given, it will be added to the Hamiltonian. So specifying {`(i,j)`:Ω_{ij}, `(j,i)`:Ω_{ij}^*} will simply double count Ω_{ij}. Default values of ε_i and Ω_{ij} are zero.

The interaction Hamiltonian (5) `coulomb` can be specified using a dictionary as

```
coulomb = {(m,n,k,l): U_mnkl,
           ...
           (i,j,p,r): U_ijpr}
```

where $U_{mnkl}d_m^\dagger d_n^\dagger d_k d_l$ and $U_{ijpr}d_i^\dagger d_j^\dagger d_p d_r$ represent some non-zero Coulomb interaction matrix elements.

The variable `nleads` is an integer and sets the number of lead channels (we will simply call it leads). The properties of leads can be given in dictionaries `mulst` (chemical potential) and `tlst` (temperature), which have `nleads` entries. Also the leads are labeled by integers from 0 to `nleads`−1. So the lead properties can be set by using

```
mulst = {0: μ_0, 1: μ_1, ..., nleads-1: μ_nleads-1}
tlst  = {0: T_0, 1: T_1, ..., nleads-1: T_nleads-1}
dband = D
```

We note that `dband` is a floating point number representing the bandwidth D of the leads.

Lastly, the single-particle tunneling amplitudes `tleads` can be specified using a dictionary as

```
tleads = {(α,i): t_αi,
          ...
          (β,j): t_βj}
```

where first index in (α, i) is the lead label and second index is the single-particle state label. Here $\mathtt{t}_{\alpha i} = \sqrt{\nu_F}t_{\alpha i}$.

[3] For the *2vN* approach it is necessary to set `kpnt`, which is the number of the equidistant energy grid points for the lead electrons.

Once the `system` object was constructed using the `Builder` class it can be solved by simply using [4]

```
system.solve()
```

The `system.solve()` performs the following steps

1. `qdq`, obtains many-body eigenstates and eigenenergies of the quantum dot Hamiltonian (3) (sets `system.qd.Ea`).
2. `rotateq`, expresses the tunnelling Hamiltonian (6) in many-body eigenbasis (sets `system.lead.Tba`).
3. `masterq`, solves approximate master equation and obtains reduced density matrix elements $\Phi^{[0]}$ (sets `system.phi0`).
4. `currentq`, calculates the current amplitudes and currents (sets `system.phi1`, `system.current`, etc.).

Unnecessary steps can be skipped. For example,

```
system.solve(qdq=True, rotateq=True, masterq=False, currentq=False)
```

will perform step **1.** (`qdq=True`) and **2.** (`rotateq=True`), however, the master equation will not be solved (`masterq=False`) and the current will not be calculated (`currentq=False`).[5]

How step **3.** is performed depends on the master equation used. For first-order approaches we solve the following linear equations

$$\mathcal{L}\Phi^{[0]} = 0, \tag{18}$$

$$\mathrm{Tr}[\Phi^{[0]}] = 1, \tag{19}$$

where $\mathcal{L}$ is the *kernel* (or *Liouvillian*) of the approach (see Appendices for more details). The solution procedure for the *2vN* approach is more complicated and is described in Appendix B.

After the calculation different properties of the system can be accessed through

```
system.current          # Particle current
system.energy_current   # Energy current
system.heat_current     # Heat current
system.phi0             # Reduced density matrix elements
system.phi1             # Energy integrated current amplitudes
system.Ea               # Eigenenergies of the quantum dot
system.Tba              # Many-body tunneling amplitudes
```

For example, `system.current` is a one-dimensional `NumPy` array of length `nleads`.

3.1. Attributes and functions of the `Builder` *class*

Here we concisely describe some of the useful attributes and functions of the `Builder` class. The variable

```
system.indexing # Possible values: 'Lin', 'charge', 'sz', 'ssq'
```

determines the labeling of states and symmetries, which can be used to simplify the numerical calculations. We note that `'sz'` stands for classification using total spin projection S_z and `'ssq'` stands for S^2. For `'Lin'` and `'charge'` only the charge is considered as a good quantum number. For now the usage of symmetries to optimize the calculations are available only for the first-order approaches in `QmeQ`. The central idea of labeling of many-body states is so-called Lin tables [48, 51]. Here Fock states are labeled by a decimal number and an actual Fock state is obtained by converting the decimal label to a binary number. For example, let us say we have `nsingle=6`, which leads to `nmany=64`. Then the labels '33' and '44' would represent different Fock states for different `indexing`

	`'Lin'`	`'charge'`	`'sz'` or `'ssq'`
33 :	$\|1,0,0,0,0,1\rangle$,	$\|1,0,0,1,0,1\rangle$,	$\|0_\uparrow,1_\uparrow,1_\uparrow,0_\downarrow,1_\downarrow,0_\downarrow\rangle$
44 :	$\|1,0,1,1,0,0\rangle$,	$\|0,1,1,0,1,1\rangle$,	$\|1_\uparrow,0_\uparrow,0_\uparrow,1_\downarrow,1_\downarrow,1_\downarrow\rangle$

[4] For the *2vN* approach it is also mandatory to specify the number of iterations, `system.solve(niter=`*integer*`)`.

[5] For the *2vN* approach, if `masterq=True` then the current is always calculated.

Decimal numbers 33 and 44 are the binary numbers 100001 and 101100. For `'charge'` indexing the Lin labels are sorted by increasing charge of the states and for `'sz'` and `'ssq'` the sorting goes first by the charge and then by the S_z value of states.

The quantum dot Hamiltonian has `nmany`=$2^{\texttt{nsingle}}$ many-body eigenstates $|b\rangle$, which in Eq. (8) are labeled by integers b from 0 to `nmany`-1. To get information about the many-body eigenstate $|b\rangle$ use the function

```
system.print_state(b)
```

and to get information about the all many-body eigenstates use

```
system.print_all_states(filename)
```

where `filename` is a string giving the file destination where this information will be printed. After diagonalisation depending on `indexing`, the many-body eigenstates have a particular labeling convention. For example, with `indexing='ssq'`, which has the highest symmetry, the states are sorted in the following order of parameters: charge, S_z, S^2, and energy E_b. To change the sorting order and relabel the states use the function

```
system.sort_eigenstates(srt=[l_0,l_1,l_2,l_3])
```

Here $l_i \in \{0,1,2,3\}$, where different numbers represent such properties: $0-E_b$, $1-$charge, $2-S_z$, $3-S^2$. So the default sorting for `'ssq'` is `srt=[1,2,3,0]`. For example, in order to sort by the energy and then by the charge use `srt=[0,1]`. However, we note that `sort_eigenstates` affects the labels when printing the states and the default labels are used in actual calculations. For instance, the order of energies in `system.Ea` or tunneling amplitudes `system.Tba` will not be changed by using `sort_eigenstates`.

In some master equation calculations not all many-body states may be relevant and contribute to the transport, because they are very far above the ground state in energy. To neglect the states with energy higher than ΔE above the ground state use

```
system.remove_states(ΔE)
```

and to reset the usage of all states call

```
system.use_all_states()
```

To access particular matrix elements of the reduced density matrix $\Phi^{[0]}_{bb'}$ and current amplitudes $\Phi^{[1]}_{cb,\alpha}$ the following function can be used

```
system.get_phi0(b, b')
system.get_phi1(α, b, b')
```

Values of $\Phi^{[0]}_{bb'}$ and $\Phi^{[1]}_{cb,\alpha}$ are stored in the arrays

```
system.phi0
system.phi1
```

We note that there is a difference how the reduced density matrix $\Phi^{[0]}$ is stored in `phi0` for the first-order approaches and the *2vN* approach. For the *2vN* approach `phi0` is a one-dimensional array of **`complex`** numbers of length `ndm0`, where `ndm0` denotes the total number of matrix elements in $\Phi^{[0]}$. For the first-order approaches we exploit the Hermiticity $\Phi^{[0]}_{b'b} = \Phi^{[0]*}_{bb'}$ and store just the elements with $b \leq b'$, where `ndm0` now denotes the number of complex elements in the upper triangle of $\Phi^{[0]}$. In this case `phi0` is a one-dimensional array of **`float`** numbers and is of length `2*ndm0-npauli`, where `npauli` is the number of diagonal elements. The diagonal elements $\Phi^{[0]}_{bb}$ can be accessed by `phi0[0:npauli]`. The entries `phi0[npauli:ndm0]` and `phi0[ndm0:]` contain the real and imaginary parts of $\Phi^{[0]}_{bb'}$ with $b < b'$. The variables `npauli` and `ndm0` are contained in

```
system.si.npauli
system.si.ndm0
```

The Liouvillian $\mathcal{L}$ corresponding to the reduced density matrix $\Phi^{[0]}$ can be accessed through

```
system.kern
```

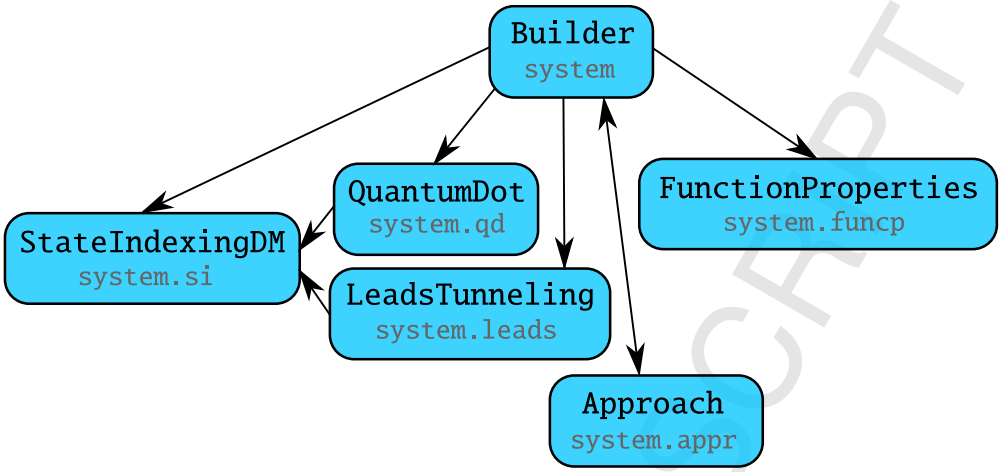


Figure 2: Important classes and their interrelation in `QmeQ` package. The direction of arrow $A \rightarrow B$ means that class A is dependent on class B.

Lastly, when doing calculations there is a need to change the parameters of the system. This can be achieved by using the functions

```
system.add(hsingle, coulomb, tleads, mulst, tlst)
system.change(hsingle, coulomb, tleads, mulst, tlst)
```

where as the names suggest `system.add` and `system.change` adds and changes the value of a parameter. For example, to change the temperature of the lead 0 to T_0 one can use the call: `system.change(tlst={0:`T_0`})`. To modify `dband` to value D simply use `system.dband=`D.

3.2. Important classes

`QmeQ` `Builder` interacts with different classes to set up the system and perform calculations using different approaches as shown in Figure 2. Here we concisely describe these classes. `StateIndexingDM` is a class describing the indexing of many-body states and density matrix elements. An object of this class is accessed in previously constructed `Builder` object `system` through

```
system.si
```

The class `StateIndexingDMc` is used for the *2vN* approach. Both `StateIndexingDM` and `StateIndexingDMc` are derived from `StateIndexing`. The `QuantumDot` is a class used internally in the `Builder` to construct and diagonalise the many-body Hamiltonian (3). An object of this class is accessed through

```
system.qd
```

The `LeadsTunneling` is a class used to represent properties of the leads, construct many-body tunneling amplitudes, and express it the eigenbasis of the Hamiltonian. An object of this class is accessed through

```
system.leads
```

The `FunctionProperties` is a class containing miscellaneous variables used by the `Approach` class. An object of this class is accessed through

```
system.funcp
```

Finally, the classes which deal with implementation of different approaches are derived from `Approach` class, and its object is

```
system.appr
```

The appr object is created using

```
appr = Approach(sys)
```

where sys is any object, which has attributes qd, leads, si, and funcp. For more details and structure see the source code of Approach class contained in qmeq.aprclass module.

For a custom approach the user can define a custom class, which needs to be derived from Approach. For example, let us take the Python implementation of the *Lindblad* approach contained in qmeq.approach.lindblad module and define a custom class from it:

```
from qmeq.aprclass import Approach
# Makes factors used for generating Lindblad master equation kernel
from qmeq.approach.lindblad import generate_tLba
# Generates master equation kernel
from qmeq.approach.lindblad import generate_kern_lindblad
# Calculates currents
from qmeq.approach.lindblad import generate_current_lindblad
# Acts on given reduced density matrix with kernel
from qmeq.approach.lindblad import generate_vec_lindblad

class ApproachCustom(Approach):

    kerntype = 'custom_pyLindblad'
    generate_fct = staticmethod(generate_tLba)
    generate_kern = staticmethod(generate_kern_lindblad)
    generate_current = staticmethod(generate_current_lindblad)
    generate_vec = staticmethod(generate_vec_lindblad)
```

We note that here we use static methods in order to have the same behavior between Python and Cython implementations. Then this custom approach can be used by setting Builder(..., kerntype=ApproachCustom). Here we used already existing implementation of the Lindblad approach, but the user can redefine, for example, the function generate_tLba, which generates particular matrix elements of the jump operators.

3.3. Parallelization

In the present version QmeQ 1.0 the modules, which generate the Hamiltonian, the Liouvillian, and calculate the current, natively do not support parallelization. However, on computers with many cores the QmeQ code still can exploit parallelization through NumPy, when calculating matrix products, the eigenvalues, or matrix inverse, which is the most time consuming for larger systems. For NumPy to have the best multithreading capabilities it should be linked to so-called ATLAS/OpenBLAS/MKL libraries. The usage of many threads can be achieved by setting an environment variable OMP_NUM_THREADS. In Python this is done in the following way (here we use 8 threads):

```
import os
os.environ['OMP_NUM_THREADS'] = '8'
```

The above method may not always work in all operating systems. It is also possible to change OMP_NUM_THREADS using the terminal of a particular operating system. For example, on Windows terminal use set OMP_NUM_THREADS=8 and on Unix/Linux/MacOS terminal use export OMP_NUM_THREADS=8. Lastly, when the calculations are made for large set of parameters it is always possible to use very simple poor-man's parallelization, where for every parameter value a new Python instance is run. This works when every instance is not too much memory consuming.

4. Example 1: Spinful single orbital

In this section we elaborate more on the spinful single orbital example (see Eq. (17) and Figure 1) and show how to calculate a stability diagram for such a quantum dot. Here we parameterize the spin-up and spin-down energies as $\varepsilon_\uparrow = V_g + \frac{B}{2}$, $\varepsilon_\downarrow = V_g - \frac{B}{2}$, where V_g is the gate voltage and B is the magnetic field (representing anomalous Zeeman splitting of spinful orbital). First, we import all the relevant prerequisites

```
# Prerequisites
import matplotlib.pyplot as plt
import numpy as np
import qmeq
```

and we choose such values for parameters:

```
# Quantum dot parameters
vgate, bfield, omega, U = 0.0, 0.0, 0.0, 20.0
# Lead parameters
vbias, temp, dband = 0.5, 1.0, 60.0
# Tunneling amplitudes
gam = 0.5
t0 = np.sqrt(gam/(2*np.pi))
```

Here the variable `gam` represents $\Gamma = 2\pi|\mathtt{t}_0|^2$ with $\mathtt{t}_0 = \sqrt{\nu_F} t_0$ (see Section 8). In H_{one} (17) we have two single particle states $|\uparrow\rangle$ and $|\downarrow\rangle$, which we label by 0 and 1, respectively. We also have four energy integrated lead channels $L\uparrow$, $R\uparrow$, $L\downarrow$, $R\downarrow$, which we label 0, 1, 2, 3. This system is constructed and solved with the *Pauli* master equation in the following way

```
nsingle = 2
# 0 is up, 1 is down
hsingle = {(0,0): vgate+bfield/2,
           (1,1): vgate-bfield/2,
           (0,1): omega}

coulomb = {(0,1,1,0): U}

tleads = {(0,0): t0, # L, up   <-- up
          (1,0): t0, # R, up   <-- up
          (2,1): t0, # L, down <-- down
          (3,1): t0} # R, down <-- down
                     # lead label, lead spin <-- level spin

nleads = 4
#        L,up        R,up         L,down      R,down
mulst = {0: vbias/2, 1: -vbias/2, 2: vbias/2, 3: -vbias/2}
tlst =  {0: temp,    1: temp,     2: temp,    3: temp}
system = qmeq.Builder(nsingle, hsingle, coulomb,
                      nleads, tleads, mulst, tlst, dband,
                      kerntype='Pauli')
system.solve()
```

We can check the obtained current and energy current using

```
print('Pauli current:')
print(system.current)
print(system.energy_current)
```

```
Pauli current:
[ 0.0207255 -0.0207255 0.0207255 -0.0207255]
[ 1.73557597e-09 -1.73557597e-09 1.73557597e-09 -1.73557597e-09]
```

The four entries correspond to current in the four lead channels $L\uparrow$, $R\uparrow$, $L\downarrow$, $R\downarrow$. Currents through the left lead and the right lead channels are conserved up to numerical errors:

```
print('Current continuity:')
print(np.sum(system.current))
```

```
Current continuity:
2.77555756156e-17
```

If we want to change the approximate approach we could redefine the system with `qmeq.Builder` by specifying the new `kerntype`. It is also possible just to change the value of `system.kerntype`:

```
kernels = ['Redfield', '1vN', 'Lindblad', 'Pauli']
for kerntype in kernels:
    system.kerntype = kerntype
    system.solve()
    print(kerntype, ' current:')
    print(system.current)
```

```
Redfield current:
[ 0.0207255 -0.0207255 0.0207255 -0.0207255]
1vN current:
[ 0.0207255 -0.0207255 0.0207255 -0.0207255]
Lindblad current:
[ 0.0207255 -0.0207255 0.0207255 -0.0207255]
Pauli current:
[ 0.0207255 -0.0207255 0.0207255 -0.0207255]
```

For the considered system, the *Pauli*, *Redfield*, *1vN*, and *Lindblad* methods all yield the same results as for given parameter values spin is a good quantum number and no coherences between $|\uparrow\rangle$ and $|\downarrow\rangle$ are developed.

In order to use the *2vN* approach, we will rebuild the system, because the solution method is rather different.[6] We also have to specify an equidistant energy grid on which the *2vN* calculations are performed (iterative solution of an integral equation):

```
kpnt = np.power(2,12)
system2vN = qmeq.Builder(nsingle, hsingle, coulomb,
                         nleads, tleads, mulst, tlst, dband,
                         kerntype='2vN', kpnt=kpnt)
```

Let us solve the *2vN* integral equations using 7 iterations:

```
system2vN.solve(niter=7)
print('Particle current:')
print(system2vN.current)
print('Energy current:')
print(system2vN.energy_current)
```

```
Particle current:
[ 0.01595739 -0.01595739 0.01595739 -0.01595739]
Energy current:
[ 0.00599524 -0.00599523 0.00599524 -0.00599523]
```

We see that the particle current is reduced and the energy current is increased considerably compared to the *Pauli* result. This is related to the inclusion of broadening in the *2vN* approach. We can check the current for every iteration to see if it has converged:

[6] The solution procedure is implemented by a `Approach2vN` object and not a `Approach` object, which is used for the first-order approaches.

```
for i in range(system2vN.niter+1):
    print(i, system2vN.iters[i].current)
```

```
0 [ 0.01505524 -0.01505524 0.01505524 -0.01505524]
1 [ 0.01589457 -0.01589457 0.01589457 -0.01589457]
2 [ 0.01595427 -0.01595427 0.01595427 -0.01595427]
3 [ 0.01595720 -0.01595720 0.01595720 -0.01595720]
4 [ 0.01595738 -0.01595738 0.01595738 -0.01595738]
5 [ 0.01595739 -0.01595739 0.01595739 -0.01595739]
6 [ 0.01595739 -0.01595739 0.01595739 -0.01595739]
```

In experiments usually the bias V and gate voltage V_g are controlled. A contour plot of current or conductance in the (V, V_g) plane is called a stability diagram. Let us produce such stability diagram for our spinful single orbital quantum dot using the *Pauli* master equation. Thus we define a function for calculation

```
def stab_calc(system, bfield, vlst, vglst, dV=0.0001):
    vpnt, vgpnt = vlst.shape[0], vglst.shape[0]
    stab = np.zeros((vpnt, vgpnt))
    stab_cond = np.zeros((vpnt, vgpnt))
    #
    for j1 in range(vgpnt):
        system.change(hsingle={(0,0):vglst[j1]+bfield/2,
                               (1,1):vglst[j1]-bfield/2})
        system.solve(masterq=False)
        for j2 in range(vpnt):
            system.change(mulst={0: vlst[j2]/2, 1: -vlst[j2]/2,
                                 2: vlst[j2]/2, 3: -vlst[j2]/2})
            system.solve(qdq=False)
            stab[j1, j2] = (system.current[0]
                          + system.current[2])
            #
            system.add(mulst={0: dV/2, 1: -dV/2,
                              2: dV/2, 3: -dV/2})
            system.solve(qdq=False)
            stab_cond[j1, j2] = (system.current[0]
                               + system.current[2]
                               - stab[j1, j2])/dV
    return stab, stab_cond
```

and a function for plotting

```
def stab_plot(stab_cond, vlst, vglst, U, gam, title):
    (xmin, xmax, ymin, ymax) = np.array([vglst[0], vglst[-1],
                                         vlst[0], vlst[-1]])/U
    fig = plt.figure(figsize=(6,4.2))
    p = plt.subplot(1, 1, 1)
    p.set_xlabel('$V_g/U$', fontsize=20)
    p.set_ylabel('$V/U$', fontsize=20)
    p.set_title(title, fontsize=20)
    p_im = plt.imshow(stab_cond.T, extent=[xmin, xmax, ymin, ymax],
                                   aspect='auto',
                                   origin='lower',
                                   cmap=plt.get_cmap('Spectral'))
    cbar = plt.colorbar(p_im)
```

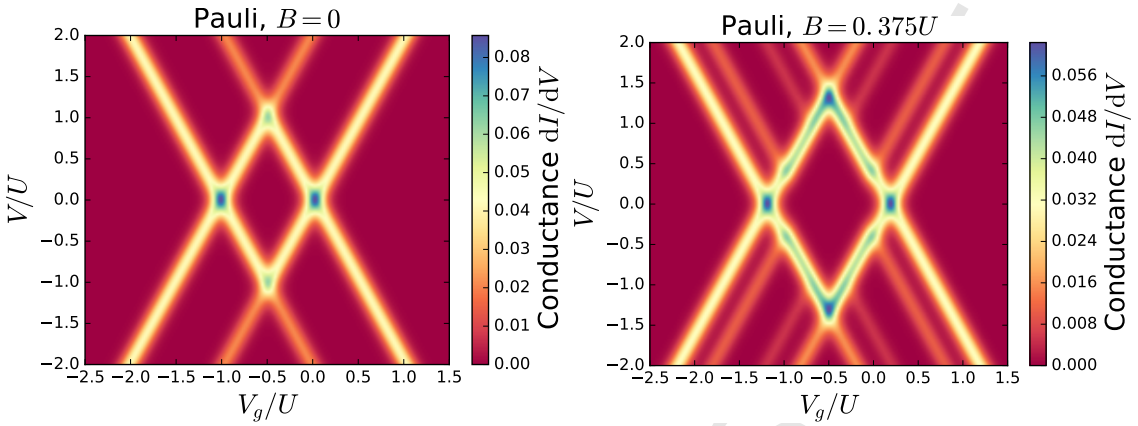


Figure 3: Stability diagrams calculated using the *Pauli* approach for a spinful single orbital quantum dot without ($B = 0$) and with ($B = 0.375$) magnetic field. Other parameters are $\Gamma_L = \Gamma_R = T/2$, $U = 20T$, $D = 60T$.

```
    cbar.set_label('Conductance dI/dV', fontsize=20)
    plt.show()
```

In `stab_calc` we changed the single-particle Hamiltonian by calling the function `system.change` and specifying which matrix elements to change. The function `system.add` adds a value to a specified parameter. Also the option `masterq=False` in `system.solve` indicates just to diagonalise the quantum dot Hamiltonian, but not to solve the master equation. Similarly, the option `qdq=False` means that the quantum dot Hamiltonian is not diagonalized (it was already diagonalized previously) and just master equation is solved. Let us now produce the stability diagram

```
system.kerntype = 'Pauli'
vpnt, vgpnt = 201, 201
vlst = np.linspace(-2*U, 2*U, vpnt)
vglst = np.linspace(-2.5*U, 1.5*U, vgpnt)
stab, stab_cond = stab_calc(system, bfield, vlst, vglst)
stab_plot(stab_cond, vlst, vglst, U, gam, 'Pauli, B = 0')
```

The result is shown in the left plot of Figure 3.

If a Zeeman splitting of the orbital is included, we obtain a stability diagram where the spin-split excited states can be seen:

```
stab_b, stab_cond_b = stab_calc(system, 7.5, vlst, vglst)
stab_plot(stab_cond_b, vlst, vglst, U, gam, 'Pauli, B = 0.375U')
```

which is shown in the right plot of Figure 3.

Now we demonstrate the difference between the *2vN* and the *Pauli* approaches by considering the bias dependence of the conductance $\mathrm{d}I/\mathrm{d}V$ at the particle-hole symmetry point $V_g = -U/2$ for $B = 0.375U$:

```
def trace_vbias(system, vlst, vgate, bfield, dV=0.01, niter=7):
    vpnt = vlst.shape[0]
    trace = np.zeros(vpnt)
    #
    system.change(hsingle={(0,0): vgate+bfield/2,
                           (1,1): vgate-bfield/2})
    system.solve(masterq=False)
    #
    for j1 in range(vpnt):
        system.change(mulst={0: vlst[j1]/2, 1: -vlst[j1]/2,
```

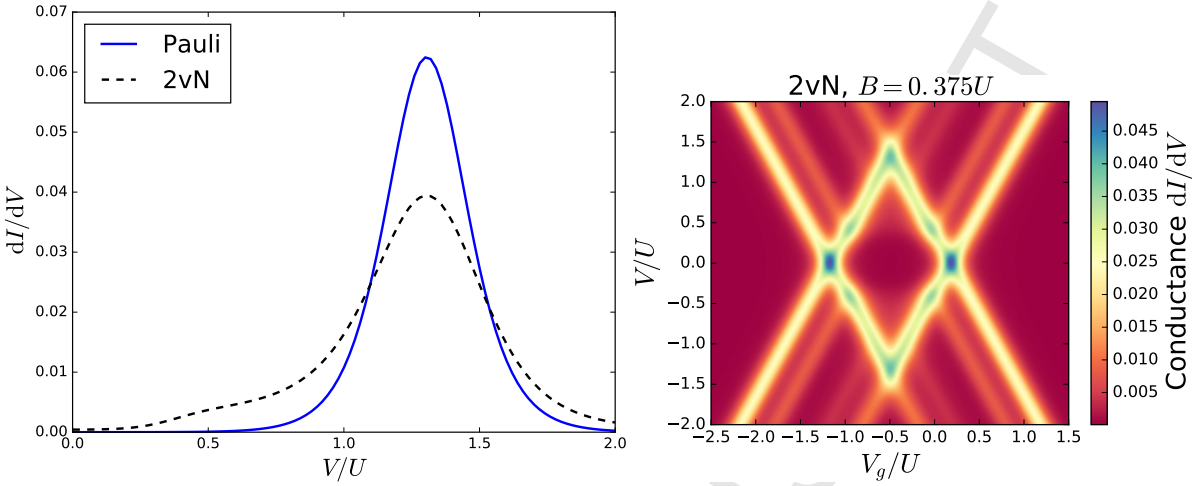


Figure 4: Left plot: bias trace at the particle-hole symmetric point $V_g = -U/2$ for $B = 0.375U$. Compared to the first-order *Pauli* master equation the *2vN* approach additionally yields a cotunneling conductance step appearing around $V = B$. Right plot: stability diagram calculated using the *2vN* approach. Other parameters are $\Gamma_L = \Gamma_R = T/2$, $U = 20T$, $D = 60T$.

```
                                2: vlst[j1]/2, 3: -vlst[j1]/2})
        system.solve(qdq=False, niter=niter)
        trace[j1] = (system.current[0]
                     + system.current[2])
        #
        system.add(mulst={0: dV/2, 1: -dV/2,
                          2: dV/2, 3: -dV/2})
        system.solve(qdq=False, niter=niter)
        trace[j1] = (system.current[0]
                     + system.current[2]
                     - trace[j1])/dV
    return trace

vpnt = 101
vlst = np.linspace(0, 2*U, vpnt)
trace_Pauli = trace_vbias(system, vlst, -U/2, 7.5)
trace_2vN = trace_vbias(system2vN, vlst, -U/2, 7.5)

fig = plt.figure()
p = plt.subplot(1, 1, 1)
p.set_xlabel('V/U', fontsize=20)
p.set_ylabel('dI/dV', fontsize=20)
plt.plot(vlst/U, trace_Pauli, label='Pauli',
                              color='blue',
                              lw=2)
plt.plot(vlst/U, trace_2vN, label='2vN',
                            color='black',
                            lw=2,
                            linestyle='--')
plt.legend(loc=2, fontsize=20)
plt.show()
```

The outcome is shown in the left plot of Figure 4. We see that the *2vN* approach captures the effect of cotunneling (conductance step appearing around $V = B$), while the first-order *Pauli* master equation does not describe this effect.

Finally, we calculate the stability diagram for finite magnetic field using *2vN* approach

```
vpnt, vgpnt = 201, 201
vlst = np.linspace(-2*U, 2*U, vpnt)
vglst = np.linspace(-2.5*U, 1.5*U, vgpnt)
stab_b_2vN, stab_cond_b_2vN = stab_calc(system2vN, 7.5, vlst, vglst)
stab_plot(stab_cond_b_2vN, vlst, vglst, U, gam, '2vN, B = 0.375U')
```

which is shown in the right plot of Figure 4.

5. Example 2: Spinless double quantum dot

Our second example is spinless serial double quantum dot described by the Hamiltonian (see Figure 5):

$$H_{\text{two}} = \sum_{k;\ell=L,R} \varepsilon_{\ell k} c^\dagger_{\ell k} c_{\ell k} + \sum_k (t d^\dagger_l c_{Lk} + t d^\dagger_r c_{Rk} + \text{H.c.}) + V_g (d^\dagger_l d_l + d^\dagger_r d_r) + (\Omega d^\dagger_l d_r + \text{H.c.}) + U d^\dagger_l d^\dagger_r d_r d_l \,. \tag{20}$$

As before we setup the system using `Builder`:

```
# Prerequisites
import matplotlib.pyplot as plt
import numpy as np
import qmeq

# Quantum dot parameters
vgate, omega, U = 0.0, 2.0, 5.0
# Lead parameters
vbias, temp, dband = 0.5, 2.0, 60.0
# Tunneling amplitudes
gam = 1.0
t0 = np.sqrt(gam/(2*np.pi))

nsingle = 2

hsingle = {(0,0): vgate,
           (1,1): vgate,
           (0,1): omega}
coulomb = {(0,1,1,0): U}

tleads = {(0,0): t0, # L <-- l
          (1,1): t0} # R <-- r

nleads = 2
#        L           R
mulst = {0: vbias/2, 1: -vbias/2}
tlst =  {0: temp,    1: temp}

system = qmeq.Builder(nsingle, hsingle, coulomb,
                      nleads, tleads, mulst, tlst, dband)
```

We are interested in how the current depends the gate voltage V_g and the hybridisation between the dots Ω, which we determine using the function:

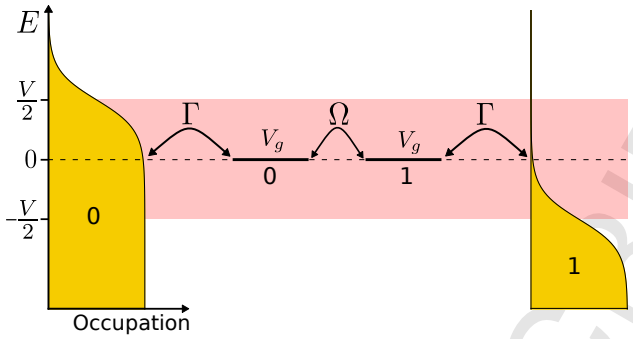


Figure 5: Spinless double quantum dot structure. The energies of the dot states are shifted by a gate voltage V_g. Both dots are coupled to each other (Ω) and to one lead each (Γ). The two leads have an applied bias voltage V, which gives a particle current I.

```
def omega_vg(system, olst, vglst):
    opnt, vgpnt = olst.shape[0], olst.shape[0]
    mtr = np.zeros((opnt, vgpnt), dtype=float)
    for j1 in range(opnt):
        system.change(hsingle={(0,1):olst[j1]})
        for j2 in range(vgpnt):
            system.change(hsingle={(0,0):vglst[j2],
                                   (1,1):vglst[j2]})
            system.solve()
            mtr[j1, j2] = system.current[0]
    return mtr
```

The plotting will be performed using:

```
def plot_omega_vg(mtr, olst, vglst, U, gam, kerntype, itype, num):
    (xmin, xmax, ymin, ymax) = (vglst[0]/U, vglst[-1]/U,
                                olst[0]/gam, olst[-1]/gam)
    p = plt.subplot(2, 2, num)
    p.set_xlabel('$V_g/\Gamma$', fontsize=20)
    p.set_ylabel('$\Omega/\Gamma$', fontsize=20)
    p.set_title(kerntype+', itype='+str(itype), fontsize=20)
    p_im = plt.imshow(mtr/gam, extent=[xmin, xmax, ymin, ymax],
                              vmin=0, vmax= 0.023,
                              aspect='auto',
                              origin='lower',
                              cmap=plt.get_cmap('Spectral'))
    cbar = plt.colorbar(p_im)
    cbar.set_label('Current [Γ]', fontsize=20)
    cbar.set_ticks(np.linspace(0.0, 0.03, 4))
    plt.tight_layout()
```

Here, for the calculations we use the *Pauli* approach and the *1vN* approach, with three different values 0, 1, and 2 for `Builder`'s optional argument `itype`. The value of `itype` determines how the integral of Eq. (C.5) is treated. In `QmeQ` for `itype=2`, the principal part $\mathcal{P}$ integrals are neglected, for `itype=1` the integrals are approximated using a digamma function [52], and for `itype=0` a full calculation is performed using the `SciPy` implementation of the `QUADPACK` routine `DQAWC`. Now we calculate the current dependence on V_g and Ω using the *Pauli* and the *1vN* methods:

```
opnt, vgpnt = 201, 201
olst = np.linspace(0., 5., opnt)
```

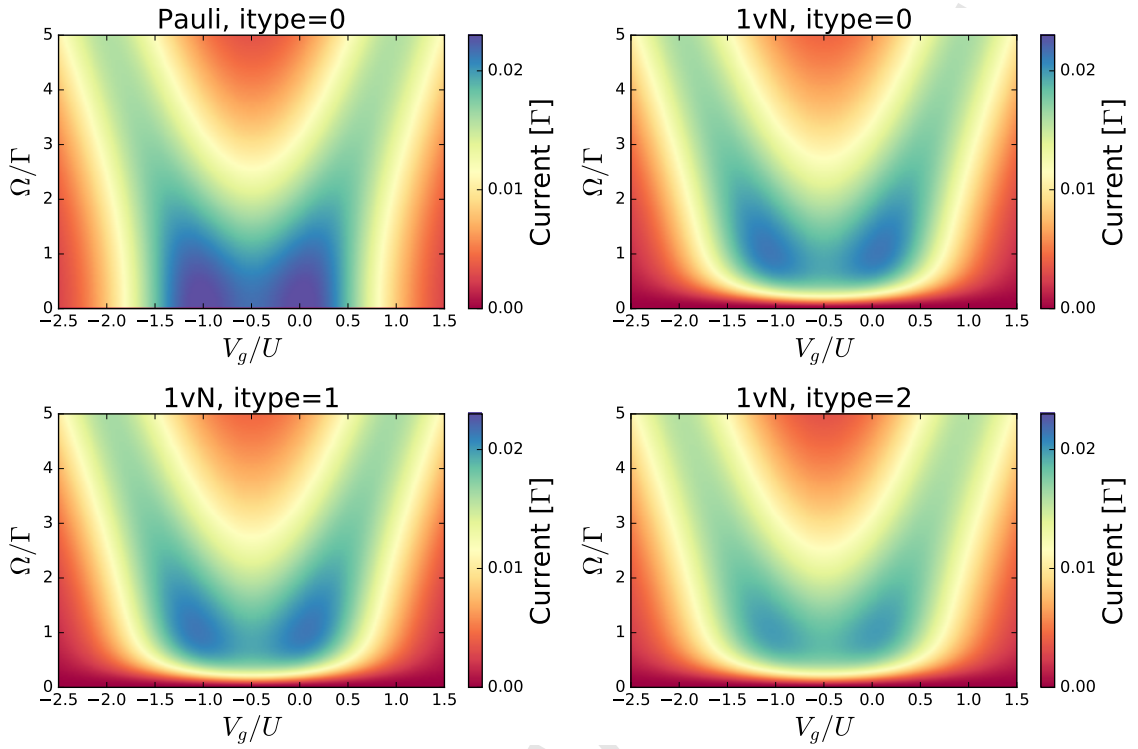


Figure 6: The dependence of the current on V_g and Ω for a spinless double quantum dot calculated using the *Pauli* and the *1vN* approaches. The values of the parameters are $\Gamma = T/2$, $U = 5T/2$, $D = 30T$.

```
vglst = np.linspace(-10., 10., vgpnt) - U/2

params = [['Pauli', 0, 1],
          ['1vN', 0, 2],
          ['1vN', 1, 3],
          ['1vN', 2, 4]]

fig = plt.figure(figsize=(12,8))

for kerntype, itype, num in params:
    system.kerntype = kerntype
    system.itype = itype
    mtr = omega_vg(system, olst, vglst)
    plot_omega_vg(mtr, olst, vglst, U, gam,
                  kerntype, itype, num)

plt.show()
```

The outcome of the calculation is shown in Figure 6. We see that the *Pauli* master equation gives the wrong current for $\Omega < \Gamma$, where the role of coherences is relevant and is correctly captured in *1vN* approach (the *Redfield*, *Lindblad*, and *2vN* approaches give similar results for the considered parameter values).

6. Example 3: Spinful serial triple-dot structure

As a last example, we consider a system with many-degrees of freedom in the quantum dot region: the spinful serial triple-dot structure shown in Figure 7. The parameters for the model are summarized in Table 1. In Ref. [34] the effect of different kinds of Coulomb matrix elements were considered in such a system. For the purposes of an example we here restrict ourselves to the dipole-dipole scattering of electrons between different quantum dots (U_{sc}). This term is responsible for the Auger process and is crucial for the current flow in the system.

Now we setup the system:

```
# Prerequisites
import matplotlib.pyplot as plt
import numpy as np
import qmeq

# Lead and tunneling parameters
mul, mur = 50.0, 10.0
gam, temp, dband = 0.1, 1.0, np.power(10.0, 4)

tl, tr = np.sqrt(gam/(2*np.pi)), np.sqrt(gam/(2*np.pi))
tleads = {(0,0): +tl, (0,1): +tl, (1,4): -tr,
          (2,5): +tl, (2,6): +tl, (3,9): -tr}

nleads = 4
mulst = {0: mul,   1: mur,   2: mul,   3: mur}
tlst  = {0: temp,  1: temp,  2: temp,  3: temp}

# Quantum dot single-particle Hamiltonian
nsingle = 10
e0, e1, e2, e3, e4 = 60, 40, 38, 20, 20
o02, o03, o12, o13, o24, o34 = 0.2, 0.1, 0.1, -0.05, 0.2, 0.1

# Spin up Hamiltonian
hsingle0 = np.array([[e0,  0,   o02, o03, 0],
                     [0,   e1,  o12, o13, 0],
                     [o02, o12, e2,  0,   o24],
                     [o03, o13, 0,   e3,  o34],
                     [0,   0,   o24, o34, e4]])
# Augment the Hamiltonian to have spin up and down
hsingle = np.kron(np.eye(2), hsingle0)

# Coulomb matrix elements
usc = -0.2
coulomb = {(0,2,3,1):usc, (0,3,2,1):usc, (0,7,8,1):usc, (0,8,7,1):usc,
           (1,2,3,0):usc, (1,3,2,0):usc, (1,7,8,0):usc, (1,8,7,0):usc,
           (2,5,6,3):usc, (2,6,5,3):usc, (3,5,6,2):usc, (3,6,5,2):usc,
           (5,7,8,6):usc, (5,8,7,6):usc, (6,7,8,5):usc, (6,8,7,5):usc}

system = qmeq.Builder(nsingle, hsingle, coulomb,
                      nleads, tleads, mulst, tlst, dband,
                      indexing='ssq', kerntype='Pauli', itype=2)
```

Here we use the spin symmetry of the problem by setting `indexing='ssq'`, which is very relevant for reducing the Liouvillian size and memory consumption in the *1vN* approach calculations. Also we neglect the principal part

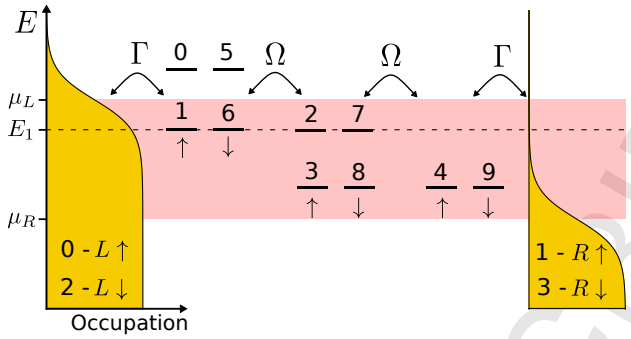


Figure 7: A spinful serial triple-dot structure studied in Ref. [34]. It has 10 single-particle states and 1024 many-body states in total.

$E_0 = E_5 = 60$	$U_{sc} = -0.2$	$\Omega_{02} = \Omega_{57} = 0.2$
$E_1 = E_6 = 40$	$\Gamma = 0.1$	$\Omega_{03} = \Omega_{58} = 0.1$
$E_2 = E_7 = 38$	$\mu_L = 50$	$\Omega_{12} = \Omega_{67} = 0.1$
$E_3 = E_8 = 20$	$\mu_R = 10$	$\Omega_{13} = \Omega_{68} = -0.05$
$E_4 = E_9 = 20$	$T = 1$	$\Omega_{24} = \Omega_{79} = 0.2$
	$D = 10^4$	$\Omega_{34} = \Omega_{89} = 0.1$

Table 1: Parameters used in the calculations of transport through a triple dot.

integrals by setting `itype=2`. Additionally, we have set up the single-particle Hamiltonian using a NumPy array.[7]

Assume we are interested in how the current depends on the level position $E_3 = E_8$, which we calculate using:

```
def trace_e3(system, e3lst, removeq=False, dE=150.0):
    e3pnt = e3lst.shape[0]
    trace = np.zeros(e3pnt)
    system.use_all_states()
    for j1 in range(e3pnt):
        system.change({(3, 3): e3lst[j1],
                       (8, 8): e3lst[j1]})
        system.solve(masterq=False)
        if removeq:
            system.remove_states(dE)
        system.solve(qdq=False)
        trace[j1] = sum(system.current[np.ix_([0,2])])
    return trace

e3pnt = 201
e3lst = np.linspace(15., 25., e3pnt)

system.kerntype = 'Pauli'
tr_e3_Pauli = trace_e3(system, e3lst)
tr_e3_Pauli_rm = trace_e3(system, e3lst, removeq=True)
system.kerntype = '1vN'
tr_e3_1vN = trace_e3(system, e3lst)
tr_e3_1vN_rm = trace_e3(system, e3lst, removeq=True)
```

[7] For more information on allowed input types in `QmeQ` see the 00_types.ipynb notebook.

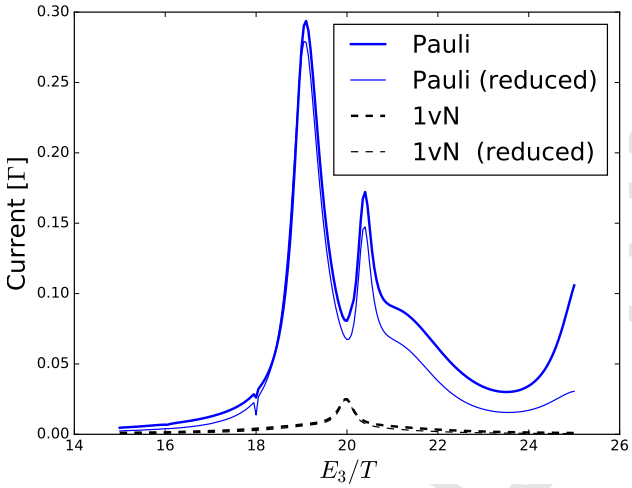


Figure 8: The dependence of the current on the level position $E_3 = E_8$ for the serial triple-dot structure. The thin curves "Pauli (reduced)" and "1vN (reduced)" correspond to the calculation when the many-body states with energy $\Delta E = 150T$ above the ground state are removed. For the *1vN* calculation this removal considerably reduces the Liouvillian size and still gives reasonable results.

```
fig = plt.figure()
p = plt.subplot(1, 1, 1)
p.set_xlabel('$E_3/T$', fontsize=20)
p.set_ylabel('Current [Γ]', fontsize=20)
plt.plot(e3lst/temp, tr_e3_Pauli/gam,
         label='Pauli', color='blue', lw=2)
plt.plot(e3lst/temp, tr_e3_Pauli_rm/gam,
         label='Pauli (reduced)', color='blue', lw=1)
plt.plot(e3lst/temp, tr_e3_1vN/gam,
         label='1vN', color='black', lw=2, linestyle='--')
plt.plot(e3lst/temp, tr_e3_1vN_rm/gam,
         label='1vN  (reduced)', color='black', lw=1, linestyle='--')
plt.legend(loc=1, fontsize=20)
plt.show()
```

The result is shown in Figure 8. We see that the current is considerably reduced in the *1vN* approach compared to the first-order *Pauli* master equation. This reduction is the effect of the coherences developing between the many-body states which have energy differences smaller than the tunneling rate Γ, similarly as discussed in the simple double-dot example in Section 5.

An important thing to note is how we used the spin symmetry (`indexing='ssq'`).[8] In order to be able to use the spin symmetry in the problem the single particle Hamiltonian, tunneling amplitudes, and the lead parameters need to be set up in a particular way. Let us say we have $n = 2m$ quantum dot single particle states counting with spin. Then we use the convention that the states with spin up have the labels $0 \ldots m-1$ and the states with spin down have the labels $m \ldots n-1$. Also there should be no coupling between the up and down states to obtain correct results. Additionally, when the spin degeneracy is present and we want to have a number m_α of physical leads, then we need to specify parameters for $n_\alpha = 2m_\alpha$ number of leads. For example, if there are source and drain leads with chemical potentials μ_L and μ_R, then we need to specify the following dictionary for chemical potentials:

```
mulst = {0: μ_L, 1: μ_R, # spin up
         2: μ_L, 3: μ_R} # spin down
```

[8] For more information on usage of symmetries in `QmeQ` see the 01_symmetries.ipynb notebook.

Now we shortly discuss how the number of $\Phi^{[0]}$ matrix elements is reduced using the spin symmetry. The many-body eigenstates $|b\rangle = |N, S, M, i\rangle$ are classified by the number of particles N, the total spin value S, and the spin-projection value M, and i denotes some other quantum numbers. In the stationary state, due to selection rules the only non-vanishing reduced density matrix $\Phi^{[0]}_{bb'}$ elements are between the states $|b\rangle = |N, S, M, i\rangle$ and $|b'\rangle = |N, S, M, i'\rangle$ for arbitrary i and i'. Additionally, all the matrix elements are equal when the states have the same N, S, i, and i' quantum numbers, e.g., $\Phi^{[0]}_{(N,S,M,i),(N,S,M,i')} = \Phi^{[0]}_{(N,S,M',i),(N,S,M',i')}$.

Lastly, to manually specify all the matrix elements related by spin symmetry can be tedious and error prone. For this reason it is possible to set up the system by giving only the values for spin-up direction and using optional argument `symmetry='spin'` in the Builder. This will automatically augment the parameters to be consistent with the spin symmetry and will use `indexing='ssq'` if it is supported by the implementation of the approach. Then for the serial triple dot considered above we can write:

```
tleads = {(0,0): +tl, (0,1): +tl, (1,4): -tr}

mulst = {0: mul,   1: mur}
tlst  = {0: temp,  1: temp}

coulomb = {(0,2,3,1):usc, (0,3,2,1):usc,
           (1,3,2,0):usc, (1,2,3,0):usc}

system = qmeq.Builder(nsingle, hsingle0, coulomb,
                      nleads, tleads, mulst, tlst, dband,
                      kerntype='Pauli', itype=2,
                      symmetry='spin')
```

We also note that if a direct interaction element like $Ud^\dagger_0 d^\dagger_5 d_5 d_0 = Ud^\dagger_{0\uparrow} d^\dagger_{0\downarrow} d_{0\downarrow} d_{0\uparrow}$ is needed and `symmetry='spin'` is used, then the user needs to specify $Ud^\dagger_0 d^\dagger_0 d_0 d_0$ in `coulomb`.

7. Computational cost

In this section we discuss the memory requirements for different approaches and computation time for examples considered in this paper. For n single-particle states we have

* $N = 2^n$ many-body states,
* $N_0 = \frac{(2n)!}{(n!)^2}$ reduced density matrix $\Phi^{[0]}$ elements,
* $N_1 = \frac{n}{n+1} N_0$ current amplitude $\Phi^{[1]}$ elements.

Here the estimate of $\Phi^{[0]}$ is done assuming that there are no coherences between the many-body states containing different number of particles, for example, $\Phi^{[0]}_{ab} = 0$, $\Phi^{[0]}_{ac} = 0$, and etc. When the Hamiltonian conserves the total number of particles such off-diagonal elements decay rapidly due to superselection rules and are not present in the stationary state [53]. Additionally, in the considered approaches when the reduced density matrix is diagonal in some conserved quantity (e. g., charge) at some point in time (e. g., in the infinite past), it will remain diagonal for all times. The estimate of $\Phi^{[1]}$ is done by counting only the matrix elements of the form $\Phi^{[1]}_{cb}$.

To store the *kernel* for the first-order approaches or the solution (with related quantities) for the *2vN* approach we need:

* $64 \times N^2$ bits for *Pauli*,
* $64 \times (2N_0 - N)^2$ bits for *1vN*, *Redfield*, *Lindblad*,
* $4 \times 128 \times N_\alpha N_k N_0 N_1$ bits for *2vN*,

where we assumed 64-bit floating point arithmetics. Here N_α is the number of lead channels and N_k is the number of energy grid points. If all many-body states are kept the memory requirement depending on the number of single-particle states n is shown in Figure 9 (for the *2vN* approach we used $N_\alpha = 2$, $N_k = 2^{12}$). For example, if 8GB of

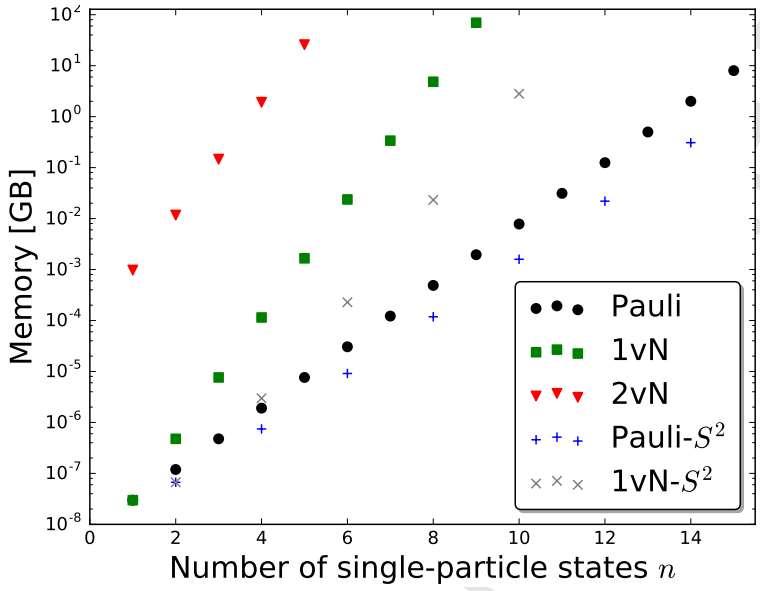


Figure 9: Memory requirement dependence on the number of single-particle states n when all many-body states are kept for the calculation. To make an estimate for the *2vN* approach we used $N_\alpha = 2$, $N_k = 2^{12}$. Pauli-S^2 and 1vN-S^2 correspond to memory requirement, when the spin symmetry is used.

memory is available then it is possible to solve a system with the following number of single-particle states: 15 for *Pauli*, 8 for *1vN*, 4 for *2vN*.[9]

The spinful serial triple-dot structure (see Section 6) with $n = 10$ single-particle states is an example, which requires a lot of memory. With all the many-body states present the spin-symmetry of the problem allows to reduce the memory requirement from 1TB to 3GB for the *1vN* approach. The calculation for a single E_3 point (see Figure 8) takes 170 seconds on our computer based on 3.50GHz processor (*Intel Xeon E3-1270v3*). For the calculation, where the many-body states with energy $\Delta E = 150T$ above the ground state are removed, the memory requirement is reduced to 0.1GB and computation time is reduced to 4 seconds.

In the spinful single orbital example (see Section 6) the *2vN* approach takes 24 seconds to calculate a single point of current on our computer with 3.50GHz processor. So it takes around 1.5 hour to produce the left plot of Figure 4 with 2×101 points of calculation and around 540 hours to produce the right plot of Figure 4 with 2×80802 points of calculation. We note that because of slow speed and large memory requirement for larger systems (see Figure 9) the *2vN* approach is useful only for describing systems with small number of many-body states.

8. Remark on units

The inputted tunneling amplitudes $\mathtt{t}_{\alpha i}$ in `tleads` correspond to tunneling amplitudes $t_{\alpha i}$ weighted by the density of states ν_F in the following way:

$$\mathtt{t}_{\alpha i} = \sqrt{\nu_F} t_{\alpha i}. \tag{21}$$

Then the tunneling rates simply become $\Gamma_{\alpha i} = 2\pi|\mathtt{t}_{\alpha i}|^2$. When different lead channels can have different density of states $\nu_{\alpha i}$, any difference can be absorbed into the tunneling amplitudes. Thus it was not necessary to specify ν_F in the given examples.

[9] We note that for the first-order approaches it is possible to use matrix-free methods for solution by specifying an optional argument `Builder(..., mfreeq=True)`. Then the memory requirement is much lower and is proportional to N or N_0 for *Pauli* or *1vN*, *Redfield*, *Lindblad* approaches, respectively. However, the computation time increases considerably.

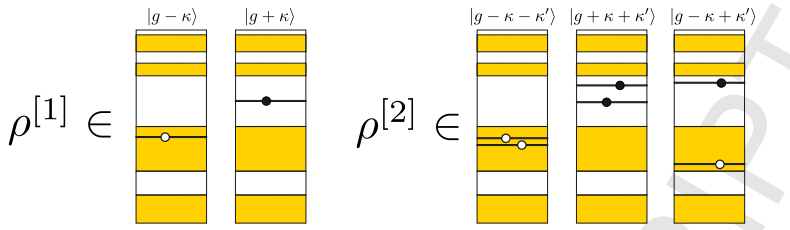


Figure A.10: Classification of the density matrix elements by the number of electron/hole excitations in the leads. $\rho^{[1]}$ or $\rho^{[2]}$ shows cases when there is a single or two electron/hole excitations in the leads, respectively. Solid dots represent electrons and hollow dots represent holes.

Throughout the calculations we have used $\hbar = 1$ and considered particle currents instead of electrical currents. If we are interested in the case $\hbar \neq 1$, $|e| \neq 1$ and in electrical currents with carriers having a charge e (which can be $e < 0$) we have to make the following changes in Figures 3–8:

$$\begin{aligned} V &\to eV, \\ V_g &\to eV_g, \\ I\ [\Gamma] &\to I\ [e\Gamma/\hbar], \\ \frac{\mathrm{d}I}{\mathrm{d}V}\ [1] &\to \frac{\mathrm{d}I}{\mathrm{d}V}\ [e^2/\hbar]. \end{aligned} \tag{22}$$

To get the conductance in units of $G_0 = e^2/h$, we should multiply the $\mathrm{d}I/\mathrm{d}V$ plots of Figures 3 and 4 by 2π.

9. Conclusion

We have presented an open-source `Python` package `QmeQ` for numerical modeling of stationary state transport through quantum dot devices, which can be described by the tunneling model (1) with quantum dots having arbitrary Coulomb interactions. The main objective of the package is to provide numerical results based on various approximate master equation approaches. We have introduced and explained the features of the package in an example driven way. The version of the package described in this paper is `QmeQ` 1.0.

Acknowledgements

We thank K. M. Seja, F. A. Damtie, B. Goldozian, and K. G. L. Pedersen for useful discussions. Financial support from the Swedish Research Council (VR) and NanoLund is gratefully acknowledged.

Appendix A. Approximate master equations

In the Appendices we present the derivation of the approximate master equations implemented in `QmeQ` and discuss their properties. These are the *second-order von Neumann* (Appendix B), the *first-order von Neumann* (Appendix C), the *first-order Redfield* (Appendix D), and the *Pauli* (Appendix E) approaches. We also shortly describe a particular form of the *Lindblad* equation in Appendix F, which takes first-order processes into consideration. It is similar to the *first-order Redfield* or *first-order von Neumann* methods, but additionally preserves positivity of the reduced density matrix [32]. In Appendix G we give suggestions in which cases which approach to use.

We are interested in solving the von Neumann equation approximately. This equation describes the evolution of the density matrix ρ of the whole system:

$$i\frac{\partial}{\partial t}\rho = [H, \rho]. \tag{A.1}$$

Our derivation of the approaches is based on the hierarchical method, where we classify the density matrix elements by the number of electron/hole excitations in the leads (see Figure A.10). Such a derivation was originally presented

in Refs. [19, 30], but most of this Appendix is based on Refs. [34, 54]. Here we derive the necessary equations to define the *second-order von Neumann* approach. Lower-order approaches will be deductively obtained in succeeding sections by making additional approximations to these equations. Density matrix elements are defined and classified as

$$\rho^{[n]}_{ag,bg'} = \langle ag|\rho|bg'\rangle, \tag{A.2}$$

where $|bg\rangle = |b\rangle \otimes |g\rangle$, with $|b\rangle$ denoting the eigenstate of the dot Hamiltonian $H_{\rm dot}$ (3) and $|g\rangle$ denoting the eigenstate of the lead Hamiltonian $H_{\rm leads}$ (2). Here the label n provides the number of electron or hole excitations needed to transform $|g\rangle$ into $|g'\rangle$. For example, we consider the matrix elements of the type

$$\begin{aligned} \rho^{[0]}_{bg,b'g} &= \langle bg|\rho|b'g\rangle, & \rho^{[2]}_{dg-\kappa-\kappa',bg} &= \langle dg-\kappa-\kappa'|\rho|bg\rangle, \\ \rho^{[1]}_{bg-\kappa,ag} &= \langle bg-\kappa|\rho|ag\rangle, & \rho^{[2]}_{bg-\kappa+\kappa',b'g} &= \langle bg-\kappa+\kappa'|\rho|b'g\rangle. \end{aligned} \tag{A.3}$$

Here we have introduced a composite index

$$\kappa \equiv k,\ \alpha; \tag{A.4}$$

and the following notation

$$\begin{aligned} |bg+\kappa\rangle &= |b\rangle \otimes c^\dagger_\kappa|g\rangle, & |dg-\kappa-\kappa'\rangle &= |d\rangle \otimes c_{\kappa'}c_\kappa|g\rangle, \\ |bg-\kappa\rangle &= |b\rangle \otimes c_\kappa|g\rangle, & |bg-\kappa+\kappa'\rangle &= |b\rangle \otimes c^\dagger_{\kappa'}c_\kappa|g\rangle. \end{aligned} \tag{A.5}$$

By neglecting all the density matrix elements with more than two electron or hole excitations $n > 2$ from Eq. (A.1) we obtain the equations

$$\begin{aligned} i\frac{\partial}{\partial t}\rho^{[0]}_{bg,b'g} = (E_b - E_{b'})\rho^{[0]}_{bg,b'g} &+ T_{ba_1,\kappa_1}\rho^{[1]}_{a_1g+\kappa_1,b'g}(-1)^{N_{a_1}} + T_{bc_1,\kappa_1}\rho^{[1]}_{c_1g-\kappa_1,b'g}(-1)^{N_b} \\ &- \rho^{[1]}_{bg,c_1g-\kappa_1}(-1)^{N_{b'}}T_{c_1b',\kappa_1} - \rho^{[1]}_{bg,a_1g+\kappa_1}(-1)^{N_{a_1}}T_{a_1b',\kappa_1}, \end{aligned} \tag{A.6}$$

$$\begin{aligned} i\frac{\partial}{\partial t}\rho^{[1]}_{cg-\kappa,bg} = (E_c - \varepsilon_\kappa - E_b)\rho^{[1]}_{cg-\kappa,bg} &+ T_{cb_1,\kappa_1}\rho^{[2]}_{b_1g-\kappa+\kappa_1,bg}(-1)^{N_{b_1}} + T_{cd_1,\kappa_1}\rho^{[2]}_{d_1g-\kappa-\kappa_1,bg}(-1)^{N_c} \\ &- \rho^{[2]}_{cg-\kappa,c_1g-\kappa_1}(-1)^{N_b}T_{c_1b,\kappa_1} - \rho^{[2]}_{cg-\kappa,a_1g+\kappa_1}(-1)^{N_{a_1}}T_{a_1b,\kappa_1}, \end{aligned} \tag{A.7}$$

$$\begin{aligned} i\frac{\partial}{\partial t}\rho^{[2]}_{bg-\kappa+\kappa',b'g} \approx (E_b - \varepsilon_\kappa + \varepsilon_{\kappa'} - E_{b'})\rho^{[2]}_{bg-\kappa+\kappa',b'g} &+ T_{ba_1,\kappa}\rho^{[1]}_{a_1g-\kappa+\kappa'+\kappa,b'g}(-1)^{N_{a_1}} + T_{bc_1,\kappa'}\rho^{[1]}_{c_1g-\kappa+\kappa'-\kappa',b'g}(-1)^{N_b} \\ &- \rho^{[1]}_{bg-\kappa+\kappa',c_1g-\kappa}(-1)^{N_{b'}}T_{c_1b',\kappa} - \rho^{[1]}_{bg-\kappa+\kappa',a_1g+\kappa'}(-1)^{N_{a_1}}T_{a_1b',\kappa'}, \end{aligned} \tag{A.8}$$

$$\begin{aligned} i\frac{\partial}{\partial t}\rho^{[2]}_{dg-\kappa+\kappa',bg} \approx (E_d - \varepsilon_\kappa - \varepsilon_{\kappa'} - E_b)\rho^{[2]}_{dg-\kappa-\kappa',bg} &+ T_{dc_1,\kappa}\rho^{[1]}_{c_1g-\kappa-\kappa'+\kappa,bg}(-1)^{N_{c_1}} + T_{dc_1,\kappa'}\rho^{[1]}_{c_1g-\kappa-\kappa'+\kappa',bg}(-1)^{N_{c_1}} \\ &- \rho^{[1]}_{dg-\kappa-\kappa',c_1g-\kappa}(-1)^{N_b}T_{c_1b,\kappa} - \rho^{[1]}_{dg-\kappa-\kappa',c_1g-\kappa'}(-1)^{N_b}T_{c_1b,\kappa'}. \end{aligned} \tag{A.9}$$

Note our convention, that we sum over all indices with subscript 1, e.g., $a_1,\ c_1,\ \kappa_1$. Additionally, phase factors like $(-1)^{N_b}$ appear due to order exchange of the lead operators with the dot operators, i.e., $c_\kappa(|b\rangle \otimes |g\rangle) = (-1)^{N_b}|b\rangle \otimes c_\kappa|g\rangle$.

Summing Eqs. (A.6) and (A.7) over all the lead states $|g\rangle$ we get

$$\begin{aligned} i\frac{\partial}{\partial t}\Phi^{[0]}_{bb'} = (E_b - E_{b'})\Phi^{[0]}_{bb'} &+ T_{ba_1,\kappa_1}\Phi^{[1]}_{a_1b',\kappa_1} + T_{bc_1,\kappa_1}\Phi^{[1]}_{c_1b',\kappa_1} \\ &- \Phi^{[1]}_{bc_1,\kappa_1}T_{c_1b',\kappa_1} - \Phi^{[1]}_{ba_1,\kappa_1}T_{a_1b',\kappa_1}, \end{aligned} \tag{A.10}$$

$$\begin{aligned} i\frac{\partial}{\partial t}\Phi^{[1]}_{cb,\kappa} \approx (E_c - \varepsilon_\kappa - E_b)\Phi^{[1]}_{cb,\kappa} &+ T_{cb_1,\kappa}\Phi^{[0]}_{b_1b}f_\kappa - \Phi^{[0]}_{cc_1}T_{c_1b,\kappa}f_{-\kappa}, + T_{cb_1,\kappa_1}\Phi^{[2]}_{b_1b,-\kappa+\kappa_1} + T_{cd_1,\kappa_1}\Phi^{[2]}_{d_1b,-\kappa-\kappa_1} \\ &+ \Phi^{[2]}_{cc_1,-\kappa+\kappa_1}T_{c_1b,\kappa_1} + \Phi^{[2]}_{ca_1,-\kappa-\kappa_1}T_{a_1b,\kappa_1}, \end{aligned} \tag{A.11}$$

$$i\frac{\partial}{\partial t}\Phi^{[2]}_{bb',-\kappa+\kappa'} \approx (E_b - \varepsilon_\kappa + \varepsilon_{\kappa'} - E_{b'})\Phi^{[2]}_{bb',-\kappa+\kappa'} - T_{ba_1,\kappa}\Phi^{[1]}_{a_1b',\kappa'}f_\kappa + T_{bc_1,\kappa'}\Phi^{[1]}_{c_1b',\kappa}f_{-\kappa'} - \Phi^{[1]}_{bc_1,\kappa'}T_{c_1b',\kappa}f_{-\kappa} + \Phi^{[1]}_{ba_1,\kappa}T_{a_1b',\kappa'}f_{\kappa'}, \tag{A.12}$$

$$i\frac{\partial}{\partial t}\Phi^{[2]}_{db,-\kappa-\kappa'} \approx (E_d - \varepsilon_\kappa - \varepsilon_{\kappa'} - E_b)\Phi^{[2]}_{db,-\kappa-\kappa'} - T_{dc_1,\kappa}\Phi^{[1]}_{c_1b,\kappa'}f_\kappa + T_{dc_1,\kappa'}\Phi^{[1]}_{c_1b,\kappa}f_{\kappa'} - \Phi^{[1]}_{dc_1,\kappa'}T_{c_1b,\kappa}f_{-\kappa} + \Phi^{[1]}_{dc_1,\kappa}T_{c_1b,\kappa'}f_{-\kappa'}, \tag{A.13}$$

where we introduced the following notation

$$\begin{aligned}
&\Phi^{[0]}_{bb'} = \sum_g \rho^{[0]}_{bg,b'g}, \quad \Phi^{[1]}_{cb,\kappa} = \sum_g \rho^{[1]}_{cg-\kappa,bg}(-1)^{N_b}, \quad \Phi^{[1]}_{bc,\kappa} = [\Phi^{[1]}_{cb,\kappa}]^*, \\
&\Phi^{[2]}_{ca,-\kappa-\kappa'} = -\sum_g \rho^{[2]}_{cg-\kappa-\kappa',ag}, \quad \Phi^{[2]}_{bb',-\kappa+\kappa'} = +\sum_g (1-\delta_{\kappa\kappa'})\rho^{[2]}_{bg-\kappa+\kappa',b'g}, \\
&f_\kappa \equiv f_{k\alpha} = (\exp[(\varepsilon_k - \mu_\alpha)/T_\alpha] + 1)^{-1}, \quad f_{-\kappa} \equiv 1 - f_{k\alpha}.
\end{aligned} \tag{A.14}$$

Going from Eq. (A.7) to Eq. (A.11) we also used

$$\begin{aligned}
\rho^{[2]}_{b_1g-\kappa+\kappa_1,bg} &= \rho^{[0]}_{b_1g-\kappa+\kappa,bg} + (1-\delta_{\kappa\kappa_1})\rho^{[2]}_{b_1g-\kappa+\kappa_1,bg}, \\
\rho^{[2]}_{cg-\kappa,c_1g-\kappa_1} &= \rho^{[0]}_{cg-\kappa,c_1g-\kappa} + (1-\delta_{\kappa\kappa_1})\rho^{[2]}_{cg-\kappa,c_1g-\kappa_1}.
\end{aligned} \tag{A.15}$$

We note that $\Phi^{[0]}_{bb'}$ represents the *reduced density matrix* of the quantum dot. Here we have also assumed that the electrons in the leads are *thermally distributed* according to the Fermi-Dirac distribution f and that this distribution is not affected by the coupling to the quantum dots. This assumption leads to the following relations for Eq. (A.7)

$$\sum_g \rho^{[0]}_{b_1g-\kappa+\kappa,bg} \approx f_\kappa \Phi^{[0]}_{b_1b}, \quad \sum_g \rho^{[0]}_{cg-\kappa,c_1g-\kappa} \approx f_{-\kappa}\Phi^{[0]}_{cc_1}, \tag{A.16}$$

for Eq. (A.8)

$$\begin{aligned}
&\sum_g \rho^{[1]}_{a_1g-\kappa+\kappa'+\kappa,b'g}(-1)^{N_{a_1}} \approx -f_\kappa \Phi^{[1]}_{a_1b',\kappa'}, \quad \sum_g \rho^{[1]}_{c_1g-\kappa+\kappa'-\kappa',b'g}(-1)^{N_b} \approx f_{-\kappa'}\Phi^{[1]}_{c_1b',\kappa}, \\
&\sum_g \rho^{[1]}_{bg-\kappa+\kappa',c_1g-\kappa}(-1)^{N_{b'}} \approx f_{-\kappa}\Phi^{[1]}_{bc_1,\kappa'}, \quad \sum_g \rho^{[1]}_{bg-\kappa+\kappa',a_1g+\kappa'}(-1)^{N_{a_1}} \approx -f_{\kappa'}\Phi^{[1]}_{ba_1,\kappa},
\end{aligned}$$

and for Eq. (A.9)

$$\begin{aligned}
&\sum_g \rho^{[1]}_{c_1g-\kappa-\kappa'+\kappa,bg}(-1)^{N_{c_1}} \approx -\Phi^{[1]}_{c_1b,\kappa'}f_\kappa, \quad \sum_g \rho^{[1]}_{c_1g-\kappa-\kappa'+\kappa',bg}(-1)^{N_{c_1}} \approx \Phi^{[1]}_{c_1b,\kappa}f_{\kappa'}, \\
&\sum_g \rho^{[1]}_{dg-\kappa-\kappa',c_1g-\kappa}(-1)^{N_b} \approx \Phi^{[1]}_{dc_1,\kappa'}f_{-\kappa}, \quad \sum_g \rho^{[1]}_{dg-\kappa-\kappa',c_1g-\kappa'}(-1)^{N_b} \approx -\Phi^{[1]}_{dc_1,\kappa}f_{-\kappa'}.
\end{aligned} \tag{A.17}$$

Equations (A.10)-(A.13) are the central equations for deriving approximate master equation implemented in `QmeQ`.

Appendix B. Solution of the second-order von Neumann approach equations

In the *second-order von Neumann* (*2vN*) approach we use the approximations:

1. Only terms involving up to two excitations $n = 2$ are kept ($\Phi^{[3]} \to 0$).
2. A Markov approximation is made to $\Phi^{[2]}$.
3. Electrons in the leads are *thermally distributed* according to the Fermi-Dirac distribution f.

Pros (+) and cons (-) of this approach:

+ Can describe sequential tunneling, cotunneling, pair-tunneling, and broadening effects [19].
+ Yields exact currents for non-interacting systems with $H_{\text{Coulomb}} = 0$ [18, 55].
- With interactions ($H_{\text{Coulomb}} \neq 0$) the results can only be trusted up to moderate coupling strengths $\Gamma \lesssim T$. Also the temperature T needs to be larger than any Kondo temperature T_K in the system [11, 19, 56].
- Can violate the positivity of the reduced density matrix $\Phi^{[0]}$ and does not satisfy the Onsager relations [57], when $H_{\text{Coulomb}} \neq 0$. However, negative occupations are rare compared to the first-order approaches addressed below and occur only for strong couplings. Similarly, the deviations from Onsager's theorem are of third order in the rates (Γ^3) as shown in [57]. Thus these features may be used to monitor the validity of results with increasing tunnel coupling.
- Has long calculation times and large memory consumption compared to the first-order approaches .

After solving the linear inhomogeneous differential equations (A.12) and (A.13) we obtain:

$$\Phi^{[2]}_{bb',-\kappa+\kappa'}(t) = \frac{1}{i}\int_{-\infty}^{t} \mathrm{d}t'\, e^{i(\varepsilon_\kappa-\varepsilon_{\kappa'}-E_b+E_{b'}+i\eta)(t-t')}\Big(- T_{ba_1,\kappa}\Phi^{[1]}_{a_1b',\kappa'}(t')f_\kappa + T_{bc_1,\kappa'}\Phi^{[1]}_{c_1b',\kappa}(t')f_{-\kappa'} - \Phi^{[1]}_{bc_1,\kappa'}(t')T_{c_1b',\kappa}f_{-\kappa} + \Phi^{[1]}_{ba_1,\kappa}(t')T_{a_1b',\kappa'}f_{\kappa'}\Big). \tag{B.1}$$

$$\Phi^{[2]}_{db,-\kappa-\kappa'}(t) = \frac{1}{i}\int_{-\infty}^{t} \mathrm{d}t'\, e^{i(\varepsilon_\kappa+\varepsilon_{\kappa'}-E_d+E_b+i\eta)(t-t')}\Big(- T_{dc_1,\kappa}\Phi^{[1]}_{c_1b,\kappa'}(t')f_\kappa + T_{dc_1,\kappa'}\Phi^{[1]}_{c_1b,\kappa}(t')f_{\kappa'} - \Phi^{[1]}_{dc_1,\kappa'}T_{c_1b,\kappa}(t')f_{-\kappa} + \Phi^{[1]}_{dc_1,\kappa}T_{c_1b,\kappa'}(t')f_{-\kappa'}\Big). \tag{B.2}$$

Here we have added a positive infinitesimal $\eta = +0$ to ensure a proper decay of initial conditions. Here we neglect the memory by replacing t' by t in the $\Phi^{[1]}$ functions of the integrals (Markov approximation). Then the integration of the exponential factor provides the final result for $\Phi^{[2]}$ functions:

$$\begin{aligned}
\Phi^{[2]}_{bb',-\kappa+\kappa'} &= \frac{-T_{ba_1,\kappa}\Phi^{[1]}_{a_1b',\kappa'}f_\kappa + T_{bc_1,\kappa'}\Phi^{[1]}_{c_1b',\kappa}f_{-\kappa'} - \Phi^{[1]}_{bc_1,\kappa'}T_{c_1b',\kappa}f_{-\kappa} + \Phi^{[1]}_{ba_1,\kappa}T_{a_1b',\kappa'}f_{\kappa'}}{\varepsilon_\kappa - \varepsilon_{\kappa'} - E_b + E_{b'} + i\eta}, \\
\Phi^{[2]}_{db,-\kappa-\kappa'} &= \frac{-T_{dc_1,\kappa}\Phi^{[1]}_{c_1b,\kappa'}f_\kappa + T_{dc_1,\kappa'}\Phi^{[1]}_{c_1b,\kappa}f_{\kappa'} - \Phi^{[1]}_{dc_1,\kappa'}T_{c_1b,\kappa}f_{-\kappa} + \Phi^{[1]}_{dc_1,\kappa}T_{c_1b,\kappa'}f_{-\kappa'}}{\varepsilon_\kappa + \varepsilon_{\kappa'} - E_d + E_b + i\eta}.
\end{aligned} \tag{B.3}$$

We note that the Markov approximation can be made quicker by this simple substitution:

$$i\left(\frac{\partial}{\partial t}+\eta\right)\Phi^{[2]}_{bb',-\kappa+\kappa'} = 0, \quad i\left(\frac{\partial}{\partial t}+\eta\right)\Phi^{[2]}_{db,-\kappa-\kappa'} = 0. \tag{B.4}$$

The above relations are also expected to hold in the stationary state. After inserting the above expressions into Eq. (A.11), we obtain the integral equations of the *2vN* method (colors are explained below)

$$\begin{aligned}
i\frac{\partial}{\partial t}\Phi^{[1]}_{cb,\kappa} &= -(\varepsilon_\kappa - E_c + E_b + i\eta)\Phi^{[1]}_{cb,\kappa} + T_{cb_1,\kappa}f_\kappa\Phi^{[0]}_{b_1b} - \Phi^{[0]}_{cc_1}f_{-\kappa}T_{c_1b,\kappa} \\
&+ \frac{T_{cb_1,\kappa_1}\left[T_{b_1c_1,\kappa_1}f_{-\kappa_1}\Phi^{[1]}_{c_1b,\kappa} + \Phi^{[1]}_{b_1a_1,\kappa}f_{\kappa_1}T_{a_1b,\kappa_1} - T_{b_1a_1,\kappa}f_\kappa\Phi^{[1]}_{a_1b,\kappa_1} - \Phi^{[1]}_{b_1c_1,\kappa_1}f_{-\kappa}T_{c_1b,\kappa}\right]}{\varepsilon_\kappa - \varepsilon_{\kappa_1} - E_{b_1} + E_b + i\eta} \\
&+ \frac{T_{cd_1,\kappa_1}\left[T_{d_1c_1,\kappa_1}f_{\kappa_1}\Phi^{[1]}_{c_1b,\kappa} + \Phi^{[1]}_{d_1c_1,\kappa}f_{-\kappa_1}T_{c_1b,\kappa_1} - T_{d_1c_1,\kappa}f_\kappa\Phi^{[1]}_{c_1b,\kappa_1} - \Phi^{[1]}_{d_1c_1,\kappa_1}f_{-\kappa}T_{c_1b,\kappa}\right]}{\varepsilon_\kappa + \varepsilon_{\kappa_1} - E_{d_1} + E_b + i\eta} \\
&+ \frac{\left[T_{cd_1,\kappa_1}f_{-\kappa_1}\Phi^{[1]}_{d_1c_1,\kappa} + \Phi^{[1]}_{cb_1,\kappa}f_{\kappa_1}T_{b_1c_1,\kappa_1} - T_{cb_1,\kappa}f_\kappa\Phi^{[1]}_{b_1c_1,\kappa_1} - \Phi^{[1]}_{cd_1,\kappa_1}f_{-\kappa}T_{d_1c_1,\kappa}\right]T_{c_1b,\kappa_1}}{\varepsilon_\kappa - \varepsilon_{\kappa_1} - E_c + E_{c_1} + i\eta} \\
&+ \frac{\left[T_{cb_1,\kappa_1}f_{\kappa_1}\Phi^{[1]}_{b_1a_1,\kappa} + \Phi^{[1]}_{cb_1,\kappa}f_{-\kappa_1}T_{b_1a_1,\kappa_1} - T_{cb_1,\kappa}f_\kappa\Phi^{[1]}_{b_1a_1,\kappa_1} - \Phi^{[1]}_{cb_1,\kappa_1}f_{-\kappa}T_{b_1a_1,\kappa}\right]T_{a_1b,\kappa_1}}{\varepsilon_\kappa + \varepsilon_{\kappa_1} - E_c + E_{a_1} + i\eta},
\end{aligned} \tag{B.5}$$

and

$$i\frac{\partial}{\partial t}\Phi^{[0]}_{bb'} = (E_b - E_{b'})\Phi^{[0]}_{bb'} + T_{ba_1,\kappa_1}\Phi^{[1]}_{a_1b',\kappa_1} + T_{bc_1,\kappa_1}\Phi^{[1]}_{c_1b',\kappa_1} - \Phi^{[1]}_{bc_1,\kappa_1}T_{c_1b',\kappa_1} - \Phi^{[1]}_{ba_1,\kappa_1}T_{a_1b',\kappa_1}. \tag{B.6}$$

Additionally, we impose the normalisation condition for the diagonal reduced density matrix elements:

$$\sum_b \Phi^{[0]}_{bb} = 1. \tag{B.7}$$

In `QmeQ` we perform steady-state transport calculations and thus have the additional conditions

$$i\frac{\partial}{\partial t}\Phi^{[0]}_{bb'} = 0, \quad \text{and} \quad i\frac{\partial}{\partial t}\Phi^{[1]}_{cb,\kappa} = 0. \tag{B.8}$$

Now we describe the numerical procedure implemented in `QmeQ` for solving the integral equations (B.5) and (B.6) in the stationary state. This numerical procedure works well for moderate coupling strengths $\Gamma \lesssim T$ and a large bandwidth $D \gg T, \Gamma$. Using Eq. (B.8), the integral equation (B.5) can be cast in the following form

$$\Phi^{[1]}_{\kappa} = F_\kappa + \sum_{\kappa_1} K_{\kappa,\kappa_1}\Phi^{[1]}_{\kappa_1}. \tag{B.9}$$

The function F_κ corresponds to a solution of Eq. (B.5) with a local approximation, i.e., terms of the form $\Phi^{[1]}_{ba,\kappa_1}$ which have integrated energy label κ_1 are neglected (colored in red). We solve Eq. (B.9) iteratively with the zeroth iteration given by $\Phi^{[1]}_{\kappa,0} = F_\kappa$. Then the first correction is determined as $\delta\Phi^{[1]}_{\kappa,1} = \sum_{\kappa_1} K_{\kappa,\kappa_1}F_{\kappa_1}$. The higher order corrections are given by $\delta\Phi^{[1]}_{\kappa,n} = \sum_{\kappa_1} K_{\kappa,\kappa_1}\delta\Phi^{[1]}_{\kappa',n-1}$ and the solution is expressed as $\Phi^{[1]}_{\kappa} = \Phi^{[1]}_{\kappa,0} + \sum_n \delta\Phi^{[1]}_{\kappa',n}$. In these iterations we have to evaluate Hilbert transforms of the form:

$$H(\Phi^{[1]}_{\kappa}) = \frac{1}{\pi}\int_{-D}^{D}\frac{\Phi^{[1]}_{\kappa'}\mathrm{d}\varepsilon_{\kappa'}}{\varepsilon_\kappa - \varepsilon_{\kappa'} \pm i\eta} = \frac{1}{\pi}\mathcal{P}\int_{-D}^{D}\frac{\Phi^{[1]}_{\kappa'}\mathrm{d}\varepsilon_{\kappa'}}{\varepsilon_\kappa - \varepsilon_{\kappa'}} \mp i\Phi^{[1]}_{\kappa}\theta(D - |\varepsilon_\kappa|). \tag{B.10}$$

The principal value integrals are efficiently evaluated on equidistant grid with N points with a fast Fourier transform, which has complexity $O(N \log N)$ [58, 59].

Appendix C. First-order von Neumann approach

The *first-order von Neumann* (*1vN*) approach is obtained with the following approximations to Eqs. (A.10)-(A.13):

1. Only terms involving up to a single excitation $n = 1$ are kept ($\Phi^{[2]} \to 0$).
2. A Markov approximation is made to $\Phi^{[1]}$.

The properties of the approach are:

- **+** Can describe sequential tunneling in the presence of coherences.
- **-** Coupling strength has to be considerably smaller than the temperature $\Gamma \ll T$.
- **-** Can violate the positivity of the reduced density matrix $\Phi^{[0]}$ [6] and does not satisfy the Onsager relations [57, 60]. Here the same consideration hold as for the *2vN* approach, but problems occur at lower coupling strengths. In particular, the deviation from the Onsager theorem is of second order in the rates (Γ^2).

We formally integrate Eq. (A.11) (with $\Phi^{[2]}$ neglected) and obtain

$$\Phi^{[1]}_{cb,\kappa}(t) = \frac{1}{i}\int_{-\infty}^{t}\mathrm{d}t' e^{i(\varepsilon_\kappa - E_c + E_b + i\eta)(t-t')}\left(T_{cb_1,\kappa}\Phi^{[0]}_{b_1b}(t')f_\kappa - \Phi^{[0]}_{cc_1}(t')T_{c_1b,\kappa}f_{-\kappa}\right). \tag{C.1}$$

Here we have added a positive infinitesimal $\eta = +0$ to ensure a proper decay of initial conditions. After performing a *Markov* approximation in the above integral, $\Phi^{[0]}_{bb'}(t') \approx \Phi^{[0]}_{bb'}(t)$, and setting $t \to +\infty$ we get:

$$\Phi^{[1]}_{cb,\kappa} = \frac{T_{cb_1,\kappa}\Phi^{[0]}_{b_1b}f_\kappa - \Phi^{[0]}_{cc_1}T_{c_1b,\kappa}f_{-\kappa}}{\varepsilon_\kappa - E_c + E_b + i\eta}. \tag{C.2}$$

The same expression (C.2) is obtained if we use $i\partial_t \Phi^{[1]} = 0$. Combining Eqs. (A.10) and (C.2) we get the *1vN* approach equations:

$$\begin{aligned} i\frac{\partial}{\partial t}\Phi^{[0]}_{bb'} = (E_b - E_{b'})\Phi^{[0]}_{bb'} &+ \sum_{b''\alpha} \Phi^{[0]}_{bb''}\Big[\sum_a \Gamma^{\alpha}_{b''a,ab'} I^{\alpha-}_{ba} - \sum_c \Gamma^{\alpha}_{b''c,cb'} I^{\alpha+*}_{cb}\Big] \\ &+ \sum_{b''\alpha} \Phi^{[0]}_{b''b'}\Big[\sum_c \Gamma^{\alpha}_{bc,cb''} I^{\alpha+}_{cb'} - \sum_a \Gamma^{\alpha}_{ba,ab''} I^{\alpha-*}_{b'a}\Big] \\ &+ \sum_{aa'\alpha} \Phi^{[0]}_{aa'} \Gamma^{\alpha}_{ba,a'b'}[I^{\alpha+*}_{b'a} - I^{\alpha+}_{ba'}] + \sum_{cc'\alpha} \Phi^{[0]}_{cc'} \Gamma^{\alpha}_{bc,c'b'}[I^{\alpha-*}_{c'b} - I^{\alpha-}_{cb'}], \end{aligned} \tag{C.3}$$

with the normalisation condition $\sum_b \Phi^{[0]}_{bb} = 1$. In Eq. (C.3) the tunneling rate matrix Γ is defined as

$$\Gamma^{\alpha}_{ba,a'b'} = 2\pi\nu_F T_{ba,\alpha} T_{a'b',\alpha}, \tag{C.4}$$

and the following integral was introduced

$$2\pi I^{\alpha\pm}_{cb} = \int_{-D}^{D} \frac{\mathrm{d}E f\left(\pm\frac{E-\mu_\alpha}{T_\alpha}\right)}{E - E_{cb} + i\eta} = \mathcal{P}\int_{-D}^{D} \frac{\mathrm{d}E f\left(\pm\frac{E-\mu_\alpha}{T_\alpha}\right)}{E - E_{cb}} - i\pi f(\pm x^{\alpha}_{cb})\theta(D - |E_{cb}|), \tag{C.5}$$

$$\text{with} \quad E_{cb} = E_c - E_b, \quad x^{\alpha}_{cb} = \frac{E_{cb} - \mu_\alpha}{T_\alpha}, \quad f(x) = [\exp(x) + 1]^{-1}, \tag{C.6}$$

which appears after performing the k-sums using a flat density of states approximation, i. e., $\sum_k \to \nu_F \int_{-D}^{D} \mathrm{d}E$, with ν_F denoting the density of states at the Fermi level and $2D$ denoting the bandwidth of the leads. For very large bandwidth $D \to \infty$ compared to the other energy scales the principal part integral in (C.5) can be approximated using a digamma function Ψ [52] in the following way

$$\mathcal{P}\int_{-D}^{D} \frac{\mathrm{d}E f\left(\frac{E-\mu_\alpha}{T_\alpha}\right)}{E - E_{cb}} \overset{D\to\infty}{\approx} \mathrm{Re}\,\Psi\left(\frac{1}{2} + i\frac{x^{\alpha}_{cb}}{2\pi}\right) - \ln\frac{D}{2\pi T_\alpha}. \tag{C.7}$$

This is the expression used for `itype=1`.

For the steady state we have $i\partial_t \Phi^{[0]}_{bb'} = 0$ and thus we can determine $\Phi^{[0]}_{bb'}$ from Eq. (C.3) and the normalisation condition $\sum_b \Phi^{[0]}_{bb} = 1$. Lastly, the particle and energy currents can be expressed as

$$I_\alpha = -2\sum_{cb} \mathrm{Im}\Big[\sum_{b'} \Gamma^{\alpha}_{bc,cb'} I^{\alpha+}_{cb} \Phi^{[0]}_{b'b} - \sum_{c'} \Gamma^{\alpha}_{bc,c'b} I^{\alpha-}_{cb} \Phi^{[0]}_{cc'}\Big], \tag{C.8}$$

$$\dot{E}_\alpha = -2\sum_{cb} \mathrm{Im}\Big[\sum_{b'} \Gamma^{\alpha}_{bc,cb'} \tilde{I}^{\alpha+}_{cb} \Phi^{[0]}_{b'b} - \sum_{c'} \Gamma^{\alpha}_{bc,c'b} \tilde{I}^{\alpha-}_{cb} \Phi^{[0]}_{cc'}\Big], \tag{C.9}$$

where for the energy currents we introduced the following integral

$$\tilde{I}^{\alpha\pm}_{cb} = \frac{1}{2\pi}\int_{-D}^{D} \frac{\mathrm{d}E E f\left(\pm\frac{E-\mu_\alpha}{T_\alpha}\right)}{E - E_{cb} + i\eta} = E_{cb} I^{\alpha\pm}_{cb} + \frac{1}{2\pi}\int_{-D}^{D} \mathrm{d}E f\left(\pm\frac{E-\mu_\alpha}{T_\alpha}\right). \tag{C.10}$$

We emphasize that there are many ways to derive the equations referred to here as the *1vN* approach. For example, identical equations result from the real-time diagrammatic technique [61, 62, 63] with a perturbation expansion truncated at leading order in the tunnel couplings. In contrast, including next-to-leading order contributions in the real-time diagrammatic technique does not give the same result as the *2vN* approach. Actually, the *2vN* approach includes some terms up to infinite order in the tunnel couplings, which is equivalent to the resonant tunneling approximation [18] in the diagrammatic approach [55].

Appendix D. First-order Redfield approach

The *first-order Redfield* approach is based on the same approximations as the *1vN* approach to Eqs. (A.10)-(A.13). Also the *Redfield* approach has the same *pros and cons* as the *1vN* approach. The difference is in the Markov approximation, where we under the integral in Eq. (C.1) set:

$$\Phi^{[0]}_{bb'}(t') \approx e^{i(E_b - E_{b'})(t-t')}\Phi^{[0]}_{bb'}(t). \tag{D.1}$$

This gives:

$$\Phi^{[1]}_{cb,\kappa} = \frac{T_{cb_1,\kappa}\Phi^{[0]}_{b_1 b} f_\kappa}{\varepsilon_\kappa - E_c + E_{b_1} + i\eta} - \frac{\Phi^{[0]}_{cc_1} T_{c_1 b,\kappa} f_{-\kappa}}{\varepsilon_\kappa - E_{c_1} + E_b + i\eta}, \tag{D.2}$$

Also note that this corresponds to expressing $\Phi^{[0]}$ in the interaction picture under the integral in Eq. (C.1) and then performing the Markov approximation $[\Phi^{[0]}_I(t') \approx \Phi^{[0]}_I(t)]$. Using Eqs. (A.10) and (D.2) we obtain the *Redfield* approach equations:

$$\begin{aligned} i\frac{\partial}{\partial t}\Phi^{[0]}_{bb'} = (E_b - E_{b'})\Phi^{[0]}_{bb'} &+ \sum_{b''\alpha}\Phi^{[0]}_{bb''}\Big[\sum_a \Gamma^\alpha_{b''a,ab'} I^{\alpha-}_{b''a} - \sum_c \Gamma^\alpha_{b''c,cb'} I^{\alpha+*}_{cb''}\Big] \\ &+ \sum_{b''\alpha}\Phi^{[0]}_{b''b'}\Big[\sum_c \Gamma^\alpha_{bc,cb''} I^{\alpha+}_{cb''} - \sum_a \Gamma^\alpha_{ba,ab''} I^{\alpha-*}_{b''a}\Big] \\ &+ \sum_{aa'\alpha}\Phi^{[0]}_{aa'}\,\Gamma^\alpha_{ba,a'b'}[I^{\alpha+*}_{b'a'} - I^{\alpha+}_{ba}] + \sum_{cc'\alpha}\Phi^{[0]}_{cc'}\,\Gamma^\alpha_{bc,c'b'}[I^{\alpha-*}_{cb} - I^{\alpha-}_{c'b'}]. \end{aligned} \tag{D.3}$$

The definitions of Γ and I are the same as in Eqs. (C.4) and (C.5). As a result of the different Markov approximation, both the equation of motion and the currents are different from the *1vN* approach. In `QmeQ` Eq. (D.3) is solved for stationary state $i\partial_t\Phi^{[0]}_{bb'} = 0$ together with normalisation condition $\sum_b \Phi^{[0]}_{bb} = 1$.

Using Eq. (D.2) the particle and energy currents become

$$I_\alpha = -2\sum_{cb}\mathrm{Im}\Big[\sum_{b'}\Gamma^\alpha_{bc,cb'} I^{\alpha+}_{cb'}\Phi^{[0]}_{b'b} - \sum_{c'}\Gamma^\alpha_{bc,c'b} I^{\alpha-}_{c'b}\Phi^{[0]}_{cc'}\Big], \tag{D.4}$$

$$\dot{E}_\alpha = -2\sum_{cb}\mathrm{Im}\Big[\sum_{b'}\Gamma^\alpha_{bc,cb'} \tilde{I}^{\alpha+}_{cb'}\Phi^{[0]}_{b'b} - \sum_{c'}\Gamma^\alpha_{bc,c'b} \tilde{I}^{\alpha-}_{c'b}\Phi^{[0]}_{cc'}\Big]. \tag{D.5}$$

Appendix E. Pauli master equation

The *Pauli* master equation is obtained with the following approximations to Eqs. (A.10)-(A.13):

1. Only the terms involving up to single excitation $n = 1$ are kept ($\Phi^{[2]} \to 0$).
2. The Markov approximation is made to $\Phi^{[1]}$.
3. The coherences of the reduced density matrix $\Phi^{[0]}$ of the quantum dot are neglected ($\Phi^{[0]}_{bb'} = 0$ for $b \neq b'$).

The properties of the approach are:

+ Can describe sequential tunneling.
+ Preserves the positivity of the reduced density matrix $\Phi^{[0]}$ [6] and satisfies the Onsager relations [64].
- The coupling strength has to be considerably smaller than the temperature $\Gamma \ll T$.
- The energy level splitting ΔE between the states with the same charge has to be considerably smaller than the coupling strength Γ [65, 34].

The *Pauli* master equation can be obtained from the *1vN* or the *Redfield* approaches by neglecting the coherences $\Phi^{[0]}_{bb'}$, $b \neq b'$. From Eq. (C.3) or (D.3) for the populations $P_b = \Phi^{[0]}_{bb}$ we obtain the equations:

$$\begin{aligned} \frac{\partial}{\partial t}P_b = &\sum_{a\alpha}\Big[P_a\Gamma^\alpha_{a\to b} f(+x^\alpha_{ba}) - P_b\Gamma^\alpha_{b\to a} f(-x^\alpha_{ba})\Big] \\ &+ \sum_{c\alpha}\Big[P_c\Gamma^\alpha_{c\to b} f(-x^\alpha_{cb}) - P_b\Gamma^\alpha_{b\to c} f(+x^\alpha_{cb})\Big], \end{aligned} \tag{E.1}$$

where we have denoted $\Gamma^{\alpha}_{a\to b} = \Gamma^{\alpha}_{ab,ba} = \Gamma^{\alpha}_{b\to a} = \Gamma^{\alpha}_{ba,ab}$. In `QmeQ` Eq. (E.1) is solved for stationary state $\partial_t P_b = 0$ together with the normalisation condition $\sum_b P_b = 1$. Using the populations P_b the particle and energy currents are expressed as

$$I_\alpha = \sum_{ab} [P_a \Gamma^{\alpha}_{a\to b} f(+x^{\alpha}_{ba}) - P_b \Gamma^{\alpha}_{b\to a} f(-x^{\alpha}_{ba})], \tag{E.2}$$

$$\dot{E}_\alpha = \sum_{bc} [P_b \Gamma^{\alpha}_{b\to c} E_{cb} f(+x^{\alpha}_{cb}) - P_c \Gamma^{\alpha}_{c\to b} E_{cb} f(-x^{\alpha}_{cb})], \tag{E.3}$$

where E_{cb} and x^{α}_{cb} are defined in (C.6).

Appendix F. Lindblad equation

The Lindblad equation [31], which we are using in `QmeQ`, has the following form:

$$\begin{aligned} i\frac{\partial}{\partial t}\Phi^{[0]}_{bb'} = (E_b - E_{b'})\Phi^{[0]}_{bb'} &- \frac{1}{2}\sum_{b''\alpha} \Phi^{[0]}_{bb''}\Big[\sum_a L^{\alpha *}_{ab''} L^{\alpha}_{ab'} + \sum_c L^{\alpha *}_{cb''} L^{\alpha}_{cb'}\Big] \\ &- \frac{1}{2}\sum_{b''\alpha} \Phi^{[0]}_{b''b'}\Big[\sum_c L^{\alpha *}_{cb} L^{\alpha}_{cb''} + \sum_a L^{\alpha *}_{ab} L_{ab''}\Big] \\ &+ \sum_{aa'\alpha} \Phi^{[0]}_{aa'} L^{\alpha}_{ba} L^{\alpha *}_{b'a'} + \sum_{cc'\alpha} \Phi^{[0]}_{cc'} L^{\alpha}_{bc} L^{\alpha *}_{b'c'}, \end{aligned} \tag{F.1}$$

where the matrix elements of the jump operators are defined as [32]

$$L_{cb,\alpha} = \sqrt{2\pi\nu_F f(+x^{\alpha}_{cb})}\, T_{bc,\alpha}, \quad L_{bc,\alpha} = \sqrt{2\pi\nu_F f(-x^{\alpha}_{cb})}\, T_{bc,\alpha}. \tag{F.2}$$

As before we solve Eq. (F.1) for the stationary state $i\partial_t \Phi^{[0]}_{bb'} = 0$ supplemented by the normalisation condition $\sum_b \Phi^{[0]}_{bb} = 1$. Some of the properties of the approach are:

- **+** Equation (F.1) is of the first-order type in the rates Γ and can describe the sequential tunneling in the presence of coherences.
- **+** Preserves the positivity of the reduced density matrix $\Phi^{[0]}$ [31].
- **-** The coupling strength has to be considerably smaller than the temperature $\Gamma \ll T$.
- **-** Does not satisfy the Onsager relations [32]. Here, the same considerations hold as for the *2vN* approach, but the deviation from the theorem is of second order in the rates (Γ^2).
- **-** Effects, which can arise due to principal part integrals [see Eq. (C.5)] are not included.

Appendix G. Which approach to use?

Having different approaches the question "Which approach to use?" can arise. From the pros (**+**) and cons (**-**) of different approaches it is clear that if a system is small and cotunneling or pair-tunneling is under interest then the *2vN* should be used. On the other hand, the *Pauli* master equation is a reliable choice with low computational cost if only sequential tunneling is of relevance and no coherences between different states develop.

If coherences are important, while tunneling is dominated by sequential events, it is possible to use the *1vN*, *Redfield*, or *Lindblad* approaches. We recommend to use the *Lindblad* approach in the case, when the effects of principal part integrals [see Eq. (C.5)] are not important, because this approach preserves positivity. At the same time we note that the principal parts are required to catch important physics such as renormalization of energy levels in some systems [66, 67, 68, 69]. By simulating different systems we also observed that with neglected principal parts the *1vN* and the *Redfield* approaches give the same results. In Ref. [57] the deviation from the Onsager's theorem was found to be slightly larger in the *1vN* approach compared to the *Redfield* approach (with the principal parts included). However, for the stationary state, the density matrix is constant in time within the Schrödinger picture. Thus for stationary states it appears to be better to do the Markov limit in the Schrödinger picture (the *1vN* approach). So we cannot give a clear-cut answer for what case the *Redfield* or *1vN* approach is better. The clear thing is that for sufficiently weak coupling both approaches do not show substantial differences.

[1] T. Chakraborty, *Quantum Dots: A Survey of the Properties of Artificial Atoms* (Elsevier Science, Amsterdam, 1999).
[2] L. P. Kouwenhoven, C. M. Marcus, P. L. Mceuen, S. Tarucha, R. M. Westervelt, and N. S. Wingreen, *Electron Transport in Quantum Dots*, edited by L. L. Sohn, L. P. Kouwenhoven, and G. Schön (Kluwer Academic Publishers, 1997).
[3] L. Yu, Z. Keane, J. Ciszek, L. Cheng, M. Stewart, J. Tour, and D. Natelson, Phys. Rev. Lett. **93**, 266802 (2004).
[4] R. Hanson, J. R. Petta, S. Tarucha, and L. M. K. Vandersypen, Rev. Mod. Phys. **79**, 1217 (2007).
[5] A. Svilans, A. M. Burke, S. F. Svensson, M. Leijnse, and H. Linke, Physica E **82**, 34 (2016).
[6] H.-P. Breuer and F. Petruccione, *Open Quantum Systems* (Oxford University Press, Oxford, 2006).
[7] H. Grabert and M. H. Devoret, *Single charge tunneling: Coulomb blockade phenomena in nanostructures* (Plenum Press and NATO Scientific Affairs Division, New York, 1992).
[8] S. De Franceschi, S. Sasaki, J. Elzerman, W. van der Wiel, S. Tarucha, and L. P. Kouwenhoven, Phys. Rev. Lett. **86**, 878 (2001).
[9] D. Goldhaber-Gordon, H. Shtrikman, D. Mahalu, D. Abusch-Magder, U. Meirav, and M. A. Kastner, Nature **391**, 156 (1998).
[10] S. M. Cronenwett, T. H. Oosterkamp, and L. P. Kouwenhoven, Science **281**, 540 (1998).
[11] M. Pustilnik and L. Glazman, J. Phys.-Condens. Mat. **16**, R513 (2004).
[12] K. Ono, D. G. Austing, Y. Tokura, and S. Tarucha, Science **297**, 1313 (2002).
[13] F. B. Anders, Phys. Rev. Lett. **101**, 066804 (2008).
[14] Y. Meir and N. S. Wingreen, Phys. Rev. Lett. **68**, 2512 (1992).
[15] D. A. Ryndyk, R. Gutiérrez, B. Song, and G. Cuniberti, in *Energy Transfer Dynamics in Biomaterial Systems*, Vol. 93, edited by I. Burghardt, V. May, D. A. Micha, and E. R. Bittner (Springer-Verlag, Berlin, Heidelberg, 2009) pp. 213–335, arXiv:0812.3335 .
[16] Y. V. Nazarov, Physica B **189**, 57 (1993).
[17] S. A. Gurvitz and Y. S. Prager, Phys. Rev. B **53**, 15932 (1996).
[18] J. König, J. Schmid, H. Schoeller, and G. Schön, Phys. Rev. B **54**, 16820 (1996).
[19] J. N. Pedersen and A. Wacker, Phys. Rev. B **72**, 195330 (2005).
[20] C. Timm, Phys. Rev. B **77**, 195416 (2008).
[21] J. Jin, X. Zheng, and Y. Yan, J. Chem. Phys. **128**, 234703 (2008).
[22] S. Koller, M. Grifoni, M. Leijnse, and M. R. Wegewijs, Phys. Rev. B **82**, 235307 (2010).
[23] H. A. Nilsson, O. Karlström, M. Larsson, P. Caroff, J. N. Pedersen, L. Samuelson, A. Wacker, L.-E. Wernersson, and H. Q. Xu, Phys. Rev. Lett. **104**, 186804 (2010).
[24] A. K. Hüttel, B. Witkamp, M. Leijnse, M. R. Wegewijs, and H. S. J. van der Zant, Phys. Rev. Lett. **102**, 225501 (2009).
[25] A. S. Zyazin, J. W. G. van den Berg, E. A. Osorio, H. S. J. van der Zant, N. P. Konstantinidis, M. Leijnse, M. R. Wegewijs, F. May, W. Hofstetter, C. Danieli, and A. Cornia, Nano Lett. **10**, 3307 (2010).
[26] W. Pauli, *Festschrift zum 60. Geburtstage A. Sommerfeld* (Hirzel, Leipzig, 1928) Pauli master equation is discussed on p. 30.
[27] H. Bruus and K. Flensberg, *Many-Body Quantum Theory in Condensed Matter Physics* (Oxford University Press, 2004).
[28] R. K. Wangsness and F. Bloch, Phys. Rev. **89**, 728 (1953).
[29] A. G. Redfield, IBM J. Res. Dev. **1**, 19 (1957).
[30] J. N. Pedersen and A. Wacker, Physica E **42**, 595 (2010).
[31] G. Lindblad, Commun. Math. Phys. **48**, 119 (1976).
[32] G. Kiršanskas, M. Franckié, and A. Wacker, (To be published) .
[33] Y. Meir, N. S. Wingreen, and P. A. Lee, Phys. Rev. Lett. **70**, 2601 (1993).
[34] B. Goldozian, F. A. Damtie, G. Kiršanskas, and A. Wacker, Sci. Rep. **6**, 22761 (2016).
[35] Python Software Foundation. Available at `http://www.python.org`.
[36] R. Bradshaw, S. Behnel, D. S. Seljebotn, G. Ewing, *et al.*, "The Cython compiler," Available at `http://cython.org`.
[37] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, Comput. Sci. Eng. **13**, 31 (2011).
[38] For list of topical software see `http://www.scipy.org/topical-software.html` (accessed 2016-11-11).
[39] S. van der Walt, S. C. Colbert, and G. Varoquaux, Comput. Sci. Eng. **13**, 22 (2011), Available at `http://www.numpy.org`.
[40] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," Available at `http://www.scipy.org`.
[41] J. D. Hunter, Comput. Sci. Eng. **9**, 90 (2007), Available at `http://matplotlib.org`.
[42] F. Perez and B. E. Granger, Comput. Sci. Eng. **9**, 21 (2007), Available at `http://jupyter.org`.
[43] G. Brandl *et al.*, "Sphinx: Python documentation generator," Available at `http://www.sphinx-doc.org`.
[44] S. R. Bahn and K. W. Jacobsen, Comput. Sci. Eng. **4**, 56 (2002), Available at `http://wiki.fysik.dtu.dk/ase/`.
[45] C. W. Groth, M. Wimmer, A. R. Akhmerov, and X. Waintal, New J. Phys. **16**, 063065 (2014), Available at `http://kwant-project.org`.
[46] J. R. Johansson, P. D. Nation, and F. Nori, Comput. Phys. Commun. **183**, 1760 (2012).
[47] J. R. Johansson, P. D. Nation, and F. Nori, Comput. Phys. Commun. **184**, 1234 (2013), Available at `http://qutip.org`.
[48] K. G. L. Pedersen, *Theoretical Investigations Regarding Single Molecules*, Ph.D. thesis, University of Copenhagen (2013).
[49] P. W. Anderson, Phys. Rev. **124**, 41 (1961).
[50] H. Haug and A.-P. Jauho, *Quantum Kinetics in Transport and Optics of Semiconductors* (Springer, Berlin Heidelberg, 1998).
[51] H. Q. Lin, J. E. Gubernatis, H. Gould, and J. Tobochnik, Comput. Phys. **7**, 400 (1993).
[52] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions* (National Bureau of Standards, Washington, D.C., 1972).
[53] W. H. Zurek, Phys. Rev. D **26**, 1862 (1982).
[54] G. Kiršanskas, S. Hammarberg, O. Karlström, and A. Wacker, Phys. Rev. B **94**, 045427 (2016).
[55] O. Karlström, C. Emary, P. Zedler, J. N. Pedersen, C. Bergenfeldt, P. Samuelsson, T. Brandes, and A. Wacker, J. Phys. A-Math. Theor. **46**, 065301 (2013).
[56] A. C. Hewson, *The Kondo Problem to Heavy Fermions* (Cambridge University Press, 1993).
[57] K. M. Seja, G. Kiršanskas, C. Timm, and A. Wacker, Phys. Rev. B **94**, 165435 (2016).
[58] T. Frederiksen, *Inelastic electron transport in nanosystems*, Master's thesis, Technical University of Denmark (2004).
[59] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes* (Cambridge University Press, Cambridge, 2007).

[60] R. Hussein and S. Kohler, Phys. Rev. B **89**, 205424 (2014).
[61] J. König, H. Schoeller, and G. Schön, Phys. Rev. Lett. **78**, 4482 (1997).
[62] H. Schoeller, Eur. Phys. J. Spec. Top. **168**, 179 (2009).
[63] M. Leijnse and M. R. Wegewijs, Phys. Rev. B **78**, 235424 (2008).
[64] R. Alicki, Rep. Math. Phys. **10**, 249 (1976).
[65] M. G. Schultz and F. von Oppen, Phys. Rev. B **80**, 033302 (2009).
[66] B. Wunsch, M. Braun, J. König, and D. Pfannkuche, Phys. Rev. B **72**, 205319 (2005).
[67] J. N. Pedersen, B. Lassen, A. Wacker, and M. H. Hettler, Phys. Rev. B **75**, 235314 (2007).
[68] B. Sothmann and J. König, Phys. Rev. B **82**, 245319 (2010).
[69] M. Misiorny, M. Hell, and M. R. Wegewijs, Nat. Phys. **9**, 801 (2013).